



Escuela Técnica Superior de
Ingeniería Informática

TRABAJO FIN DE GRADO

Equipo para el estudio de control aplicado a sistemas biomecatrónicos

Realizado por

José Pablo Palmero Ramos

Para la obtención del título de
Grado en Ingeniería de la Salud
Mención en Ingeniería Biomédica

Dirigido por

Larios Marín, Diego Francisco
Luque Giráldez, José Rafael

Departamento de
Tecnología Electrónica

Sevilla, junio de 2024

Agradecimientos

Gracias a todos aquellos que me han estado a mi lado estos años.

A mis padres y a mi hermano por apoyarme siempre.

A mis amigos por sacarme una sonrisa cuando más me hacía falta.

A mi pareja por creer en mí como nadie.

Y en especial, a ese año en Italia que me ayudó a seguir para poder estar hoy aquí.

Resumen

El presente trabajo se centra en el diseño y desarrollo de un sistema biomecatrónico para la rehabilitación de pacientes con patologías en la mano. El objetivo principal es replicar los movimientos de rehabilitación de los pacientes a través de una mano robótica en la clínica del fisioterapeuta, permitiendo así un monitoreo en tiempo real de la evolución del paciente, combinando herramientas de software de captura de imágenes con el hardware que replica los movimientos. Este sistema busca ofrecer una solución híbrida, combinando sesiones presenciales y telemáticas, optimizando así el tiempo y los recursos tanto para los pacientes como para los fisioterapeutas. El enfoque del proyecto se centra en aquellas personas que se encuentren en lista de espera para una rehabilitación física y presencial, y en aquellos pacientes con movilidad reducida.

Palabras clave: Rehabilitación, biomecatrónica, servomotores, mano, fisioterapia, imagen.

Abstract

This work focuses on the design and development of a biomechatronic system for the rehabilitation of patients with hand pathologies. The main objective is to replicate the rehabilitation movements of patients through a robotic hand in the physiotherapist's clinic, allowing real-time monitoring of the patient's progress. This system aims to offer a hybrid solution, combining in-person and telematic sessions, thereby optimizing time and resources for both patients and physiotherapists. The project targets individuals on waiting lists for physical and in-person rehabilitation, as well as patients with reduced mobility.

Keywords: Rehabilitation, biomechatronics, servomotors, hand, physiotherapy, imaging.

Índice general

Contenido

1.	Introducción	1
1.1.	Motivación	1
1.2.	Estado del arte	1
1.3.	Justificación.....	3
1.4.	Objetivos	3
2.	Organización y planificación temporal.....	4
3.	Justificación clínica	6
3.1.	Movimiento flexión del pulgar	6
3.2.	Movimiento flexión interfalángica del pulgar	6
3.3.	Movimiento pinza índice	7
3.4.	Flexión extensión global de todos los dedos	8
4.	Diseño, materiales, recursos y entornos utilizados	10
4.1.	Diseño	10
4.2.	Materiales.....	21
4.2.1.	Materiales de impresión 3D	21
4.2.2.	Materiales principales	22
4.2.3.	Materiales montaje	23
4.3.	Recursos utilizados	23
4.4.	Entornos desarrollo	24
5.	Desarrollo	25
5.1.	Análisis de requisitos.....	25
5.2.	Desarrollo hardware	27
5.2.1.	Desarrollo	27
5.2.1.1.	Desarrollo 3D y estructural.....	27
5.2.1.2.	Desarrollo de los motores LX16A y conexión TTL.....	28
5.2.2.	Montaje y ensamblado.....	30
5.2.2.1.	Impresión 3D de las piezas.....	30
5.2.2.2.	Preparación de las piezas de la mano	31
5.2.2.3.	Montaje de las piezas de la mano	31
5.2.2.4.	Estudio de la transmisión del movimiento	34
5.2.2.5.	Colocación de los motores.....	37
5.2.3.	Resultado	39
5.3.	Desarrollo software	39
5.3.1.	Desarrollo inicial del código	39

5.3.1.1.	Búsqueda de librerías LX16A	40
5.3.1.2.	Búsqueda de recursos para la captura de gestos	41
5.3.1.3.	Configuración de los motores mediante GUI.....	43
5.3.2.	Desarrollo completo del código.....	43
5.3.2.1.	Importación.....	43
5.3.2.2.	Primer ejercicio de flexión extensión completa.....	44
5.3.2.3.	Segundo ejercicio de flexión del pulgar:.....	45
5.3.2.4.	Tercer ejercicio de flexión interfalángica del pulgar:.....	46
5.3.2.5.	Cuarto ejercicio de pinza índice:	46
5.3.2.6.	Función de restablecer posición:.....	47
5.3.2.7.	Captura de imagen y creación de lista de valores:.....	47
5.3.3.	Resultados esperados	49
6.	Cinemática del sistema.....	51
6.1.	Introducción del sistema.....	51
6.2.	Cálculo de posiciones iniciales	52
6.3.	Cálculo de velocidades.....	54
6.3.1.	Cálculo de velocidades por articulación	54
7.	Pruebas y Resultados.....	58
7.1.	Pruebas manuales	58
7.1.1.	Prueba ID de los servomotores	58
7.1.2.	Prueba posiciones servomotores	60
7.1.3.	Prueba sobre las articulaciones de los dedos	60
7.2.	Resultados	61
8.	Análisis temporal y viabilidad	61
8.1.	Análisis temporal	61
8.2.	Viabilidad y desarrollo futuro	63
9.	Conclusiones	655
10.	Referencias.....	666

Índice de figuras

- Figura 2.1: Diagrama de Gantt - Página 5
- Figura 4.1: Auriculaire.stl - Página 11
- Figura 4.2: Bolt_Entretoise7.stl - Página 12
- Figura 4.3: Index3.stl - Página 12
- Figura 4.4: Majeure3.stl - Página 13
- Figura 4.5: RingFinger3.stl - Página 13
- Figura 4.6: CoverFinger1.stl - Página 14
- Figura 4.7: robcap3V2.stl - Página 14
- Figura 4.8: robpart2V4.stl - Página 15
- Figura 4.9: robpart3V4.stl - Página 15
- Figura 4.10: robpart4V4.stl - Página 16
- Figura 4.11: robpart5V4.stl - Página 16
- Figura 4.12: thumb5.stl - Página 17
- Figura 4.13: topsurface6.stl - Página 17
- Figura 4.14: topsurfaceUP6.stl - Página 18
- Figura 4.15: WristLargeV4.stl - Página 18
- Figura 4.16: WristsmallV4.stl - Página 19
- Figura 4.17: servo-pulleyX5.stl - Página 19
- Figura 4.18: WristGearsV5.stl - Página 19
- Figura 4.19: RotaWrist1V4.stl - Página 20
- Figura 4.20: RotaWrist2V3.stl - Página 20
- Figura 4.21: RotaWrist3V3.stl - Página 21
- Figura 5.1: Conexión Servo LX16A - Página 28
- Figura 5.2: Disposición de los motores - Página 29
- Figura 5.3: Dedo completo montado - Página 32
- Figura 5.4: Extesión de la mano completa - Página 33
- Figura 5.5: Motor con la polea incorporada - Página 36
- Figura 5.6: Mano alineada con eje vertical - Página 37
- Figura 5.7: Disposición de los motores - Página 38
- Figura 5.8: Diagrama completo del sistema - Página 39

- Figura 5.9: Detección de la mano completa - Página 50
- Figura 6.1: Diagrama de bloques simplificando el dedo - Página 52
- Figura 6.2: Captura del código en LaTeX de la definición de jacobiana - Página 55
- Figura 6.3: Captura del código en LaTeX de la matriz jacobiana - Página 55
- Figura 6.4: Captura del código en LaTeX de la relación de velocidades - Página 56
- Figura 6.5: Captura de LaTeX de la relación de velocidades articulares - Página 56
- Figura 6.6: Captura de LaTeX del sistema de ecuaciones - Página 56
- Figura 6.7: Resolución del sistema de ecuaciones - Página 57
- Figura 7.1: Captura del GUI - Página 59
- Figura 7.2: Captura del GUI del cambio de ID - Página 59
- Figura 7.3: Captura del GUI del cambio de posición - Página 60
- Figura 8.1: Diagrama de Gantt real - Página 62
- Figura 8.2: Comparativa del Diagrama de Gantt real y el estimado - Página 63

Índice de extractos de código

- Código 5.1: Importación librerías - Página 44
- Código 5.2: Ejercicio flexión y extensión - Página 44
- Código 5.3: Ejercicio flexión del pulgar- Página 45
- Código 5.4: Ejercicio flexión interfalángica del pulgar- Página 46
- Código 5.5: Ejercicio pinza del índice - Página 46
- Código 5.6: Función restablecer posición de motores- Página 47
- Código 5.7: Captura de imagen - Página 47
- Código 5.8: Consola - Página 49
- Código 8.1: Métodos librería Mediapipe - Página 64

Índice de tablas

- Cuadro 2.1: Organización inicial- Página 4
- Cuadro 4.1: Especificaciones de los servomotores - Página 22
- Cuadro 4.2: Materiales empleados - Página 23
- Cuadro 5.1: Requisitos del sistema- Página 25
- Cuadro 5.2: Requisitos de sistema y de usuario- Página 27
- Cuadro 5.3: Tiempos de impresión 3D- Página 30
- Cuadro 5.4: ID de los motores - Página 38
- Cuadro 6.1: Cinemática del sistema -Página 54
- Cuadro 6.2: Velocidades angulares - Página 57
- Cuadro 6.3: Velocidades angulares -Página 57
- Cuadro 8.1: Planificación real - Página 62

1. Introducción

El proyecto realizado tiene como objetivo principal brindar un sistema de apoyo biomecatrónico a los especialistas del campo de la rehabilitación. En concreto, el sistema diseñado está enfocado en replicar aquellos movimientos que los pacientes con algún tipo de patología en la mano realizan como ejercicios de rehabilitación de manera autónoma. Esos movimientos serán replicados por una mano en la clínica del fisioterapeuta para que pueda monitorizar la evolución del paciente.

De esta forma, se implementa una herramienta visual que permite controlar la evolución en tiempo real de aquellos pacientes que tengan alguna limitación en el desplazamiento.

1.1. Motivación

En la actualidad me encuentro en una empresa dedicada al campo de la rehabilitación mediante herramientas digitales, lo que me ha permitido expandir mis conocimientos en esta área. Todo ello, combinado con los conocimientos técnicos y sanitarios adquiridos durante el grado, me ha impulsado a diseñar un sistema de apoyo a la rehabilitación que combina a la perfección los campos de ingeniería de control, electrónica y anatomía.

El poder construir una mano que tuviese como finalidad ayudar a aquellas personas que no pueden desplazarse a una consulta a realizar sus sesiones de rehabilitación era algo que me decantó por apostar por este proyecto.

1.2. Estado del arte

Para contextualizar el estado del arte del presente proyecto, se dividirá en varias partes.

En primer lugar, sobre la robótica aplicada al campo de la rehabilitación. Actualmente existen diversos robots diseñados para ayudar y facilitar en el proceso de rehabilitación a todos los pacientes posibles. Ejemplos como el robot MIT-Manus, el cual presta un servicio a pacientes que hayan sufrido accidentes cardíacos, o robots más orientados a prótesis como puede ser el robot LUKE.

Por otro lado, el uso de servomotores para el control eficiente de este tipo de robots está muy extendido. En concreto, el uso de servomotores para controlar movimientos de partes del cuerpo como la mano es óptimo, al permitir una precisión alta y evitando picos y ruido que harían menos precisos los movimientos.

Como contexto al problema que se pretende resolver, se han estudiado dos artículos que pueden servir de contexto sobre los últimos avances en la rehabilitación mediante el uso de robots.

La rehabilitación física asistida por robots ha emergido como una solución prometedora para mejorar el proceso de recuperación de los pacientes. En particular, el uso de robots humanoides como Pepper para replicar movimientos captados por cámaras y supervisar ejercicios de rehabilitación ha mostrado ser efectivo en la mejora de la adherencia y la eficacia del tratamiento. Este estado del arte analiza dos artículos clave en este campo: uno sobre el uso de robots en la terapia física después de fracturas de mano y otro sobre el desarrollo de robots de terapia física con base en bases de datos de rehabilitación.

El artículo "Towards Supervised Robot-assisted Physical Therapy after Hand Fractures" de Hrabar et al. explora el uso del robot humanoide Pepper para asistir a fisioterapeutas durante la rehabilitación de pacientes con fracturas de mano. Pepper está configurado para demostrar y supervisar dieciocho ejercicios estándar de rehabilitación. Utilizando redes neuronales, Pepper puede detectar el desempeño del paciente y proporcionar retroalimentación en tiempo real. Este sistema no solo permite que los pacientes realicen los ejercicios correctamente, sino que también ofrece motivación continua al interactuar verbalmente con ellos.

El estudio encontró que utilizando datasets de lenguajes de señas, se podía obtener una retroalimentación confiable sobre el desempeño del ejercicio, lo cual permite al robot decidir el número de repeticiones exitosas. Además, se mencionan los desafíos como la detección precisa de movimientos de la mano del paciente y las limitaciones en la movilidad de los brazos de Pepper.

El segundo artículo, "Development of a Physical Therapy Robot for Rehabilitation" de otro grupo de investigadores, se centra en la creación de una base de datos robusta para apoyar el desarrollo de robots de terapia física. Este documento describe cómo se pueden utilizar datos de diferentes fuentes para entrenar modelos de aprendizaje automático que permiten a los robots reconocer y evaluar los movimientos del paciente durante los ejercicios de rehabilitación.

Este enfoque se complementa con hardware adicional, como unidades FPGA PYNQ y Nvidia Jetson Tx2, para manejar el procesamiento de imágenes capturadas por las cámaras del robot. Esta infraestructura permite que el robot no solo demuestre los ejercicios, sino que también evalúe la precisión de los movimientos del paciente en tiempo real, proporcionando una supervisión continua sin la necesidad de intervención constante del fisioterapeuta.

La combinación de robots humanoides y sistemas de cámaras presenta varias ventajas en la rehabilitación física. La interacción continua y motivacional del robot con el paciente puede aumentar la adherencia a los ejercicios prescritos. Los robots pueden proporcionar una evaluación continua y precisa del desempeño del paciente, ajustando las repeticiones y ofreciendo retroalimentación en tiempo real. Al delegar tareas repetitivas al robot, los fisioterapeutas pueden dedicar más tiempo a pacientes con necesidades más complejas.

Por último, el uso de herramientas software para la rehabilitación (lo que da lugar al término más extendido de tele rehabilitación) se encuentra en auge actualmente. En concreto, existen empresas como Healthinn que se encargan de facilitar a los pacientes una alternativa a la terapia presencial, incentivando la rehabilitación híbrida,

acortando tiempos de espera y evitando desplazamientos innecesarios o a personas que tengan algún tipo de discapacidad para desplazarse.

Cabe añadir que herramientas basadas en inteligencia artificial y Machine Learning como la librería Mediapipe de Google potencian toda la captura de imágenes que se emplean en las terapias de tele rehabilitación.

1.3. Justificación

Al tener un problema real en el ámbito sanitario en cuanto a tiempos de espera, priorizando la terapia presencial, el construir un sistema que pudiese ayudar a promover la rehabilitación de manera híbrida era un objetivo por cumplir.

Actualmente la sociedad se encuentra con graves problemas en cuanto a tiempos de espera para poder conseguir una cita con un fisioterapeuta, pudiendo agravar la patología que se tenga al no obtener ningún tipo de tratamiento.

Todo ello, sumado a aquellos pacientes que tengan algún tipo de patología o discapacidad de movilidad y no puedan desplazarse a una clínica u hospital cercano de manera presencial, fomenta la creación de herramientas digitales que sustituyan o ayuden a la recuperación.

1.4. Objetivos

Por todo lo citado previamente, el objetivo principal de este proyecto es construir una herramienta dedicada a pacientes y fisioterapeutas que facilite la recuperación de aquellos pacientes que tengan dificultades para poder acudir de manera presencial a una consulta, y además ahorre recursos y tiempos de espera en consulta a los especialistas.

Cabe recalcar que el objetivo principal no es sustituir la terapia presencial, sino promover un modelo híbrido en el tratamiento de pacientes con consultas sucesivas.

Por todo ello, se busca que la herramienta diseñada facilite la rehabilitación, pero nunca sea un sustituyente de la terapia presencial.

Esta herramienta aumentará la capacidad de pacientes tratables por especialistas, al poder reducir el número de consultas, ya que se dividirán en sesiones presenciales y telemáticas. De esta manera, los huecos que se liberarían para sesiones sucesivas de distintos pacientes los podrían utilizar unos nuevos.

La implantación de este tipo de herramientas supondría una monitorización similar a la presencial al ser un modelo visual en tres dimensiones que replique los ejercicios que el paciente esté realizando.

2. Organización y planificación temporal

En este apartado, se resumirá cómo se han organizado las tareas del proyecto de manera estimada y se mostrarán visualmente en un Diagrama de Gantt.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Búsqueda de información sobre la temática del TFG	15 días	lun 16/10/23	vie 03/11/23	
Búsqueda de documentación sobre motores y modelo 3D	59 días?	mié 25/10/23	lun 15/01/24	
Poner en funcionamiento los motores	14 días?	lun 15/01/24	jue 01/02/24	
Estudio transmisión movimiento	13 días	jue 01/02/24	lun 19/02/24	
Impresión 3D de las piezas	30 días?	lun 19/02/24	vie 29/03/24	
Ensamblado de las piezas 3D	22 días?	lun 01/04/24	mar 30/04/24	5
Montaje estructura	22 días?	lun 01/04/24	mar 30/04/24	
Afinar funcionamientos motores	13 días?	lun 15/04/24	mié 01/05/24	
Justificación clínica	5 días?	lun 01/04/24	dom 07/04/24	
Pruebas de los motores	5 días?	jue 25/04/24	mié 01/05/24	
Cálculo de cinemática	6 días?	mié 01/05/24	mié 08/05/24	6
Memoria del proyecto	11 días?	mié 01/05/24	mié 15/05/24	7
Fin TFG	1 día	mié 15/05/24	mié 15/05/24	

Cuadro 2.1: Organización inicial

En la tabla anterior se resume la planificación estimada de las tareas usando el software Microsoft Project.

Se establece una línea base por si hubiese retrasos futuros y así poder ver la comparativa posteriormente.

La distribución temporal de las tareas quedaría de la siguiente manera:



Figura 2.1: Diagrama de Gantt

En el diagrama anterior se observan las estimaciones de cada tarea.

Un aspecto que destacar sería que la búsqueda de documentación de motores y piezas 3D es la tarea más alargada, dado que la documentación sobre los motores que se van a utilizar en el proyecto es limitada, y el diseño de una mano en 3D también. Por tanto, se estima una mayor duración en buscar una información adecuada a toda la base del proyecto.

3. Justificación clínica

Durante este apartado se justificará la validez clínica en diferentes patologías de los ejercicios propuestos para la rehabilitación.

3.1. Movimiento flexión del pulgar

Flexión Pulgar:

Descripción del ejercicio: partiendo de la posición de la mano abierta completamente, lleva la punta del dedo pulgar hacia la base del dedo meñique. A continuación, vuelve a la posición inicial y repite. El objetivo de este ejercicio es la movilidad articular del dedo pulgar, así como la mejora en el rango de movimiento.

La flexión del pulgar es un componente crucial en la rehabilitación de una mano hemipléjica, ya que el pulgar desempeña un papel fundamental en una amplia gama de actividades funcionales. La evidencia derivada de varios estudios revisados respalda la inclusión de ejercicios de flexión del pulgar en programas de rehabilitación después de fracturas del miembro superior. Estos ejercicios han demostrado ser efectivos en la mejora de la fuerza y el rango de movimiento en el pulgar, así como en la prevención de complicaciones como la rigidez articular y las adherencias tendinosas.

La aplicación temprana y activa de ejercicios específicos, como la flexión del pulgar, es fundamental para optimizar la recuperación funcional. La movilización precoz y dirigida del pulgar puede ayudar a restaurar la función muscular y articular, lo que resulta en una recuperación más rápida y completa. Además, la flexión del pulgar se centra en fortalecer los músculos intrínsecos de la mano, lo que contribuye a mejorar la destreza manual y la coordinación neuromuscular necesaria para realizar actividades cotidianas.

Según un estudio de Van de Beek (2020) demostraron que los ejercicios orientados a la destreza manual son efectivos y bien aceptados en poblaciones con problemas neurológicos. La alta adherencia observada en el estudio (97%) sugiere que los pacientes están dispuestos a seguir programas de ejercicio, especialmente cuando estos son accesibles y fáciles de seguir.

3.2. Movimiento flexión interfalángica del pulgar

Flexión Interfalángica del Pulgar:

Descripción del ejercicio: se flexiona y se extiende el pulgar intentando que solo se mueva la última falange. El objetivo de este ejercicio es la movilidad articular de la articulación interfalángica del pulgar, así como la activación de toda la musculatura que lo involucra.

Este ejercicio se dirige específicamente a mejorar la movilidad y la fuerza en las articulaciones del pulgar, lo que es crucial para restaurar la funcionalidad de la mano afectada. La literatura científica respalda la inclusión de ejercicios de flexión interfalángica del pulgar en programas de rehabilitación, ya que estos ejercicios pueden prevenir la atrofia muscular y mejorar la coordinación mano-ojo.

Los estudios revisados han demostrado que la incorporación de ejercicios específicos como la flexión interfalángica del pulgar en programas de rehabilitación contribuye significativamente a la recuperación funcional. Este tipo de ejercicio no solo mejora la movilidad y la fuerza en las articulaciones del pulgar, sino que también promueve la estabilidad y la funcionalidad de la mano en su conjunto. Además, la flexión interfalángica del pulgar puede mejorar la coordinación mano-ojo y la destreza manual, lo que facilita la realización de tareas cotidianas.

Patologías:

Efectos del trabajo domiciliario en fracturas de metacarpo: Los programas de ejercicio domiciliario son efectivos en el tratamiento postoperatorio de fracturas metacarpianas. La flexión interfalángica del pulgar es fundamental para mejorar la movilidad y la fuerza de la mano afectada, lo que contribuye a una recuperación exitosa.

Prevención de complicaciones: La realización de ejercicios específicos, como la flexión interfalángica del pulgar, puede prevenir complicaciones como la adherencia cicatricial y la contractura muscular, favoreciendo así la rehabilitación completa de la mano hemipléjica.

Por último, según el mismo estudio anterior de Van de Beek (2020), entrenar en casa es altamente factible para pacientes con condiciones neurológicas, ofreciendo flexibilidad en la organización de horarios y ahorro de tiempo.

3.3. Movimiento pinza índice

Pinza Índice:

Descripción del ejercicio: se realiza el gesto de la pinza en apertura y en cierre utilizando el pulgar y el dedo índice. El objetivo de este ejercicio es la movilidad articular de los dedos, además de la coordinación y movilidad fina de la mano. Es un ejercicio muy trasladable a las actividades de la vida cotidiana.

La pinza índice es un ejercicio específico que se enfoca en mejorar la capacidad de agarre, una habilidad esencial en la rehabilitación de la mano hemipléjica u otras patologías neurológicas. Este ejercicio implica la coordinación de múltiples músculos y articulaciones para realizar un agarre preciso y eficaz. La evidencia derivada de varios estudios respalda la inclusión de ejercicios de agarre, como la pinza índice, en programas de rehabilitación, ya que estos ejercicios fortalecen los músculos de la mano y mejoran la coordinación mano-ojo.

Los estudios revisados han demostrado que la aplicación de ejercicios de agarre, como la pinza índice, contribuye significativamente a la recuperación funcional y la independencia del paciente. Este tipo de ejercicio no solo mejora la fuerza de agarre,

sino que también promueve la estabilidad y la coordinación de la mano, lo que facilita la realización de actividades cotidianas y laborales.

Comparación entre terapia presencial y domiciliaria: Estudios sugieren que la terapia presencial y el trabajo domiciliario producen resultados similares en la rehabilitación de fracturas de radio distal. La inclusión de ejercicios como la pinza índice en el programa de ejercicios domiciliarios mejora la destreza manual y la funcionalidad de la mano.

Promoción de la adherencia al tratamiento: La prescripción de ejercicios específicos, como la pinza índice, en el entorno domiciliario aumenta la participación activa del paciente en su proceso de recuperación y garantiza una mayor adherencia al tratamiento, lo que conduce a mejores resultados a largo plazo.

Estos ejercicios, individualmente o en combinación, son fundamentales para mejorar la movilidad, la fuerza y la funcionalidad de la mano, contribuyendo así a una rehabilitación exitosa y a una mejor calidad de vida del paciente.

3.4. Flexión extensión global de todos los dedos

Descripción del ejercicio: se flexionan todos los dedos todo lo posible cerrando la mano en posición de puño. A continuación, se abre la mano extendiendo los dedos todo lo que se pueda. El objetivo de este ejercicio es la movilidad articular de todos los dedos, así como la coordinación y la mejora en el rango de movimiento.

El artículo de Harris et al. (2009) destaca la importancia de programas de rehabilitación autoadministrados para mejorar la funcionalidad del brazo en pacientes con accidente cerebrovascular (ACV). El programa GRASP (Graded Repetitive Arm Supplementary Program) es un enfoque que ha demostrado ser efectivo en la rehabilitación de la extremidad superior. Este programa se puede adaptar para incluir ejercicios específicos de flexión y extensión de los dedos, lo cual es crucial para la recuperación de la función manual.

La inclusión temprana en programas de rehabilitación mejora significativamente la recuperación funcional en pacientes con ACV.

Aumentar la intensidad del tratamiento mediante tareas repetitivas orientadas funcionalmente mejora el desempeño motor de la extremidad superior.

Por otro lado, los programas autoadministrados de ejercicio, como GRASP, son viables y costo-efectivos, ya sea de forma aislada o en combinación con el tratamiento hospitalario. Además, los pacientes muestran un alto grado de adherencia y satisfacción con estos programas.

El programa GRASP establece tres niveles de dosificación de los ejercicios, acorde a la escala FMA-UE, que se corresponden con una discapacidad leve, moderada o severa. Los ejercicios incluyen actividades de fuerza, destreza fina y rango de movimiento.

En concreto, en lo que concierne al ejercicio, se concluye con que: los ejercicios deben repetirse durante 3 series, con un número de repeticiones que oscila entre 5 (nivel inicial, dificultad baja) y 10 (nivel más demandante, dificultad alta). Además, para trabajar la parte distal de la extremidad superior, se necesita al menos 10º de

extensión activa de muñeca y dedos contra la gravedad. Los ejercicios pueden generar sensación de cansancio muscular y deben suponer un reto sin implicar un aumento no tolerable del dolor, y ante la presencia de signos de entumecimiento, el paciente debe descansar y disminuir el nivel de dificultad.

Para lograr un mayor sentido funcional, es necesario involucrar ambas manos en tareas bimanuales y alternar su uso durante algunos ejercicios. La progresión de los ejercicios debe basarse en el porcentaje de aciertos y errores en la ejecución de la tarea o en la imposibilidad de completar un número mínimo de repeticiones.

Un programa de ejercicios desarrollado de forma autónoma en población con ACV es seguro y no se han reportado efectos adversos serios asociados. El nivel de fatiga se mide con la Fatigue Severity Scale y la funcionalidad del miembro superior con la escala ARAT.

Por último, el programa GRASP y ejercicios similares de flexión y extensión de los dedos son beneficiosos para varias patologías. Para pacientes con accidente cerebrovascular (ACV), la rehabilitación temprana y la intensificación de ejercicios funcionales son esenciales para la recuperación. Para aquellos con debilidad muscular, estos ejercicios mejoran la fuerza y destreza fina. En casos de rigidez y espasticidad, la repetición de ejercicios puede ayudar a reducir la rigidez y mejorar el rango de movimiento. Finalmente, para el deterioro funcional de la mano, el uso de tareas bimanuales y el aumento progresivo de la dificultad mejoran la coordinación y funcionalidad de la mano.

4. Diseño, materiales, recursos y entornos utilizados

4.1. Diseño

Para el diseño de la mano se ha hecho uso de la plataforma InMoov. Se ha hecho uso del diseño de la mano derecha para este proyecto.

InMoov es una plataforma de distribución de software libre de diseño 3D enfocado en el diseño de robots humanoides.

La plataforma permite extraer diseños de cada parte del cuerpo humano en formato STL, el cual almacena la información 3D para su posterior impresión. Además, permite la impresión por separado de cada una de las piezas, además de instrucciones detalladas de su ensamblado.

Por otro lado, posee una comunidad activa de usuarios a través del enlace de la plataforma Thingiverse, donde se permite compartir problemas y soluciones en el ensamblado y montaje del diseño. Por último, InMoov ofrece una compatibilidad con diferentes plataformas de software y hardware como son Arduino o MyRobotLab.

Por todo ello, el diseño de la mano derecha de la plataforma InMoov fue el implementado en el proyecto, permitiendo obtener unas piezas optimizadas.

Posteriormente, se ha visualizado el diseño en el software Fusion 360, el cual permite ajustar y comprobar las medidas de cada pieza para poder ensamblarlas una vez impresas.

Fusion 360 es un software de diseño asistido por computadora desarrollado por Autodesk, diseñado principalmente para piezas de robots en 3D. Combina diseño industrial y mecánico, colaboración en equipo y fabricación, facilitando el trabajo a lo largo del proceso de desarrollo del producto. Fusion 360 permite crear modelos paramétricos, definiendo y ajustando dimensiones y características de las piezas de manera precisa y eficiente. También cuenta con ensamblajes, diseñar piezas individuales del robot y luego ensamblarlas para asegurar el posicionamiento correcto y la funcionalidad conjunta. Está integrado con herramientas de fabricación asistida por computadora (CAM), permitiendo la prueba en tiempo real, simulación, análisis de tensiones y deformaciones, y validación de la durabilidad del diseño antes de la fabricación.

Fusion 360 exporta diseños a formatos compatibles con impresoras 3D o máquinas CNC, facilitando la creación de prototipos y la producción del producto final.

Las principales ventajas de Fusion 360 para el diseño de piezas de robots incluyen la integración completa de diseño, simulación y fabricación, accesibilidad en la nube, colaboración en tiempo real y su versatilidad para soportar diversos procesos de diseño y fabricación.

A continuación, se adjuntan todas las piezas con sus medidas en milímetros que se han impreso y utilizado en el diseño de la mano:

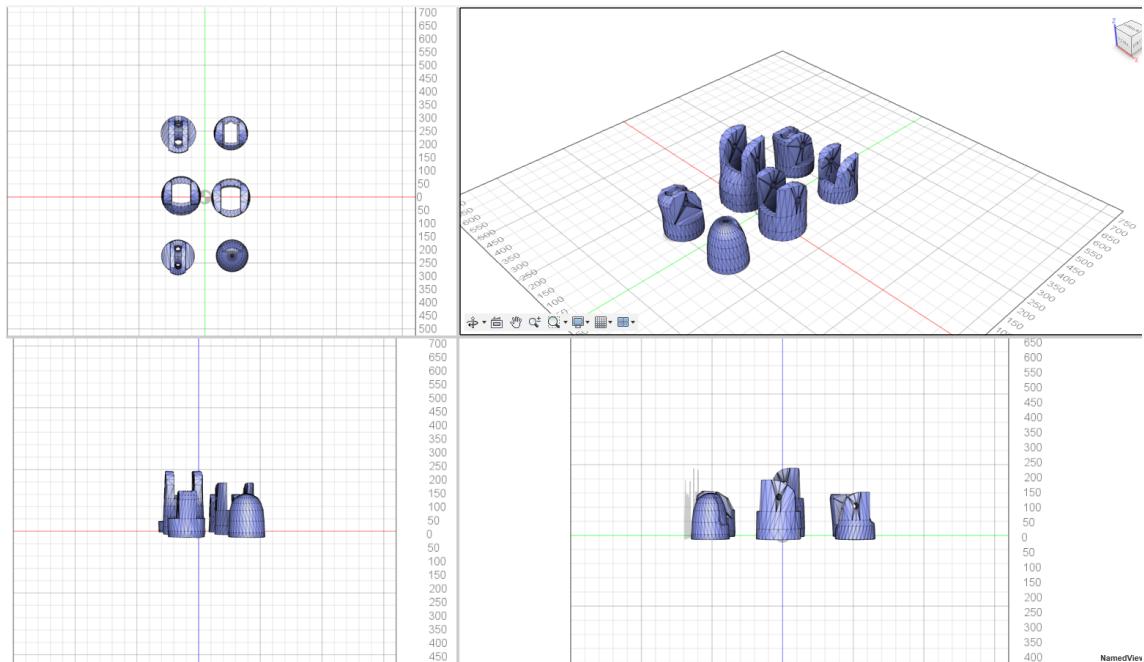


Figura 4.1: Auriculaire.stl

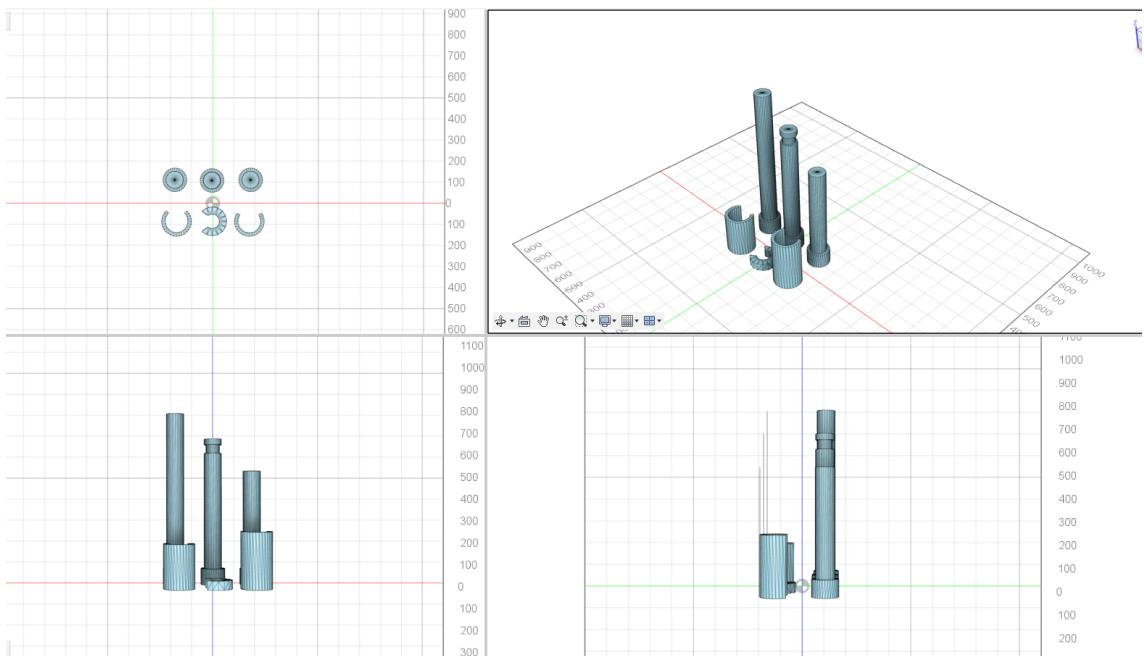


Figura 4.2: Bolt_Entretoise7.stl

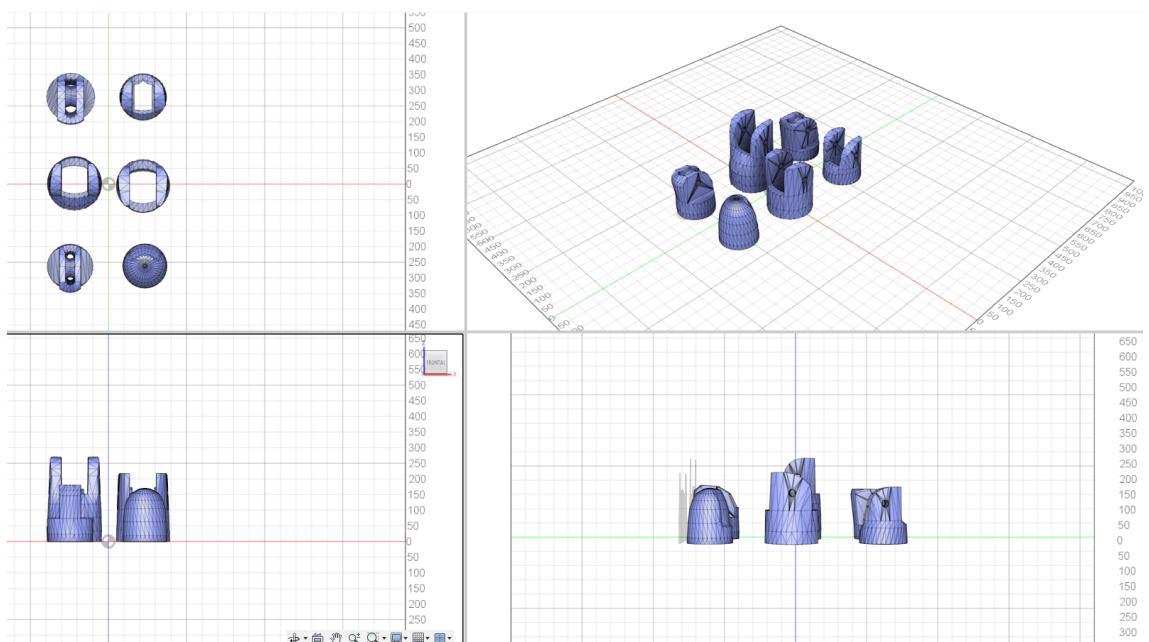


Figura 4.3: Index3.stl

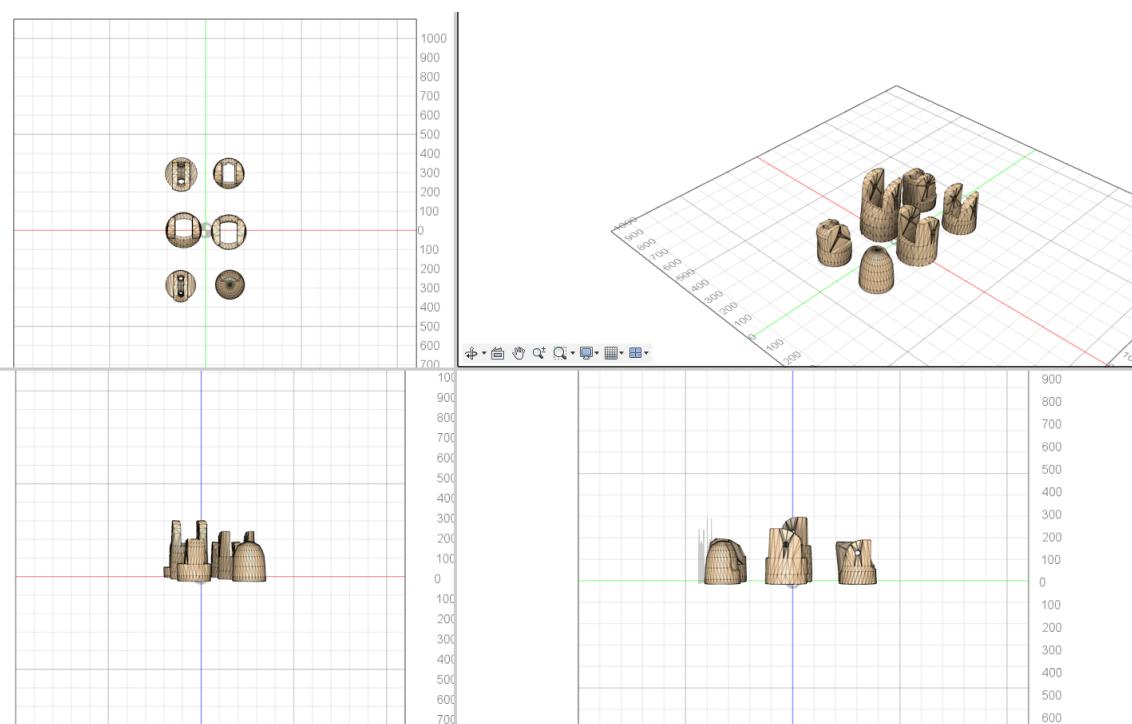


Figura 4.4: Majeure3.stl

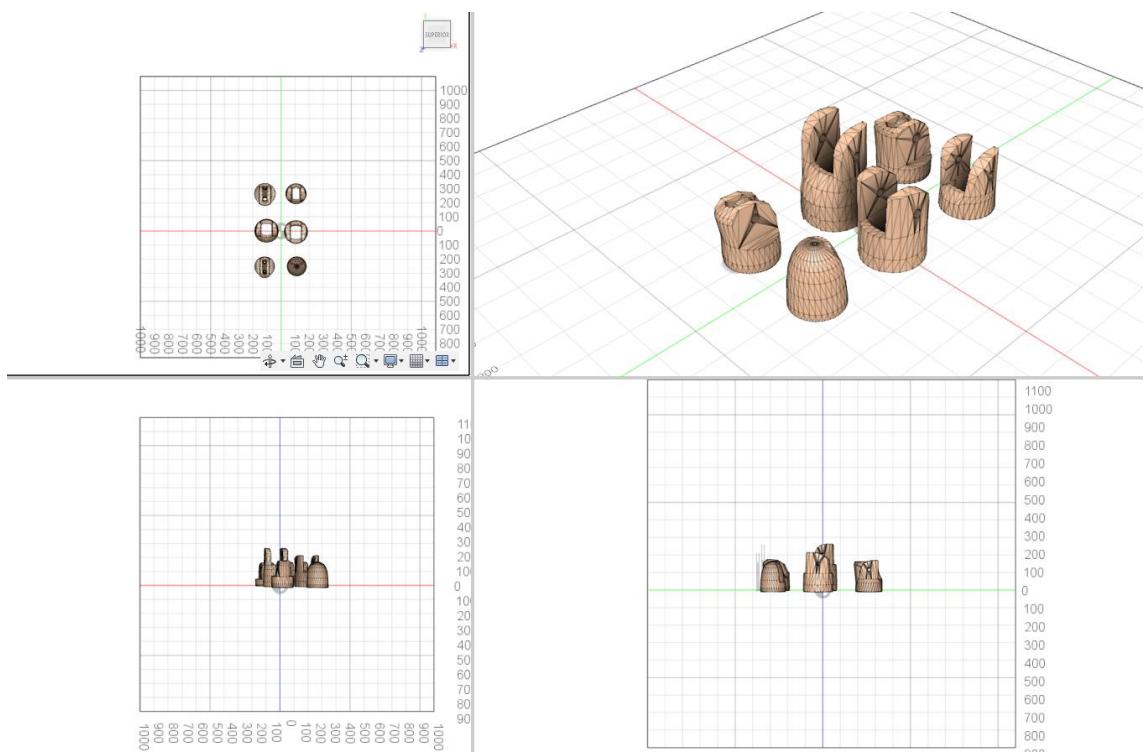


Figura 4.5: RingFinger3.stl

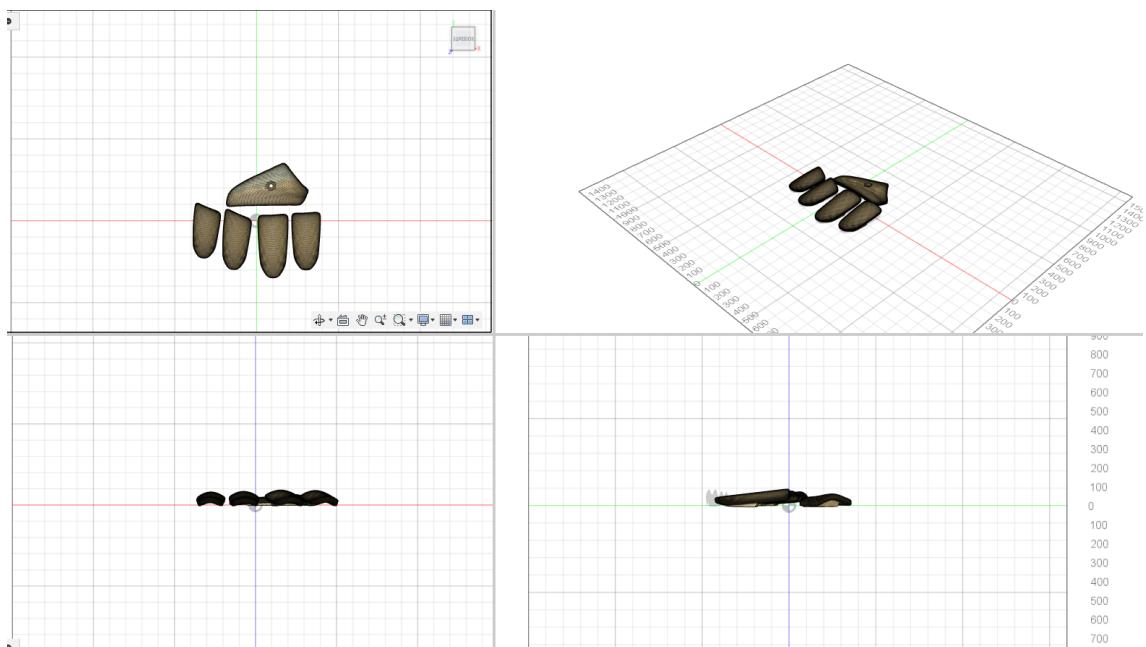


Figura 4.6: CoverFinger1.stl

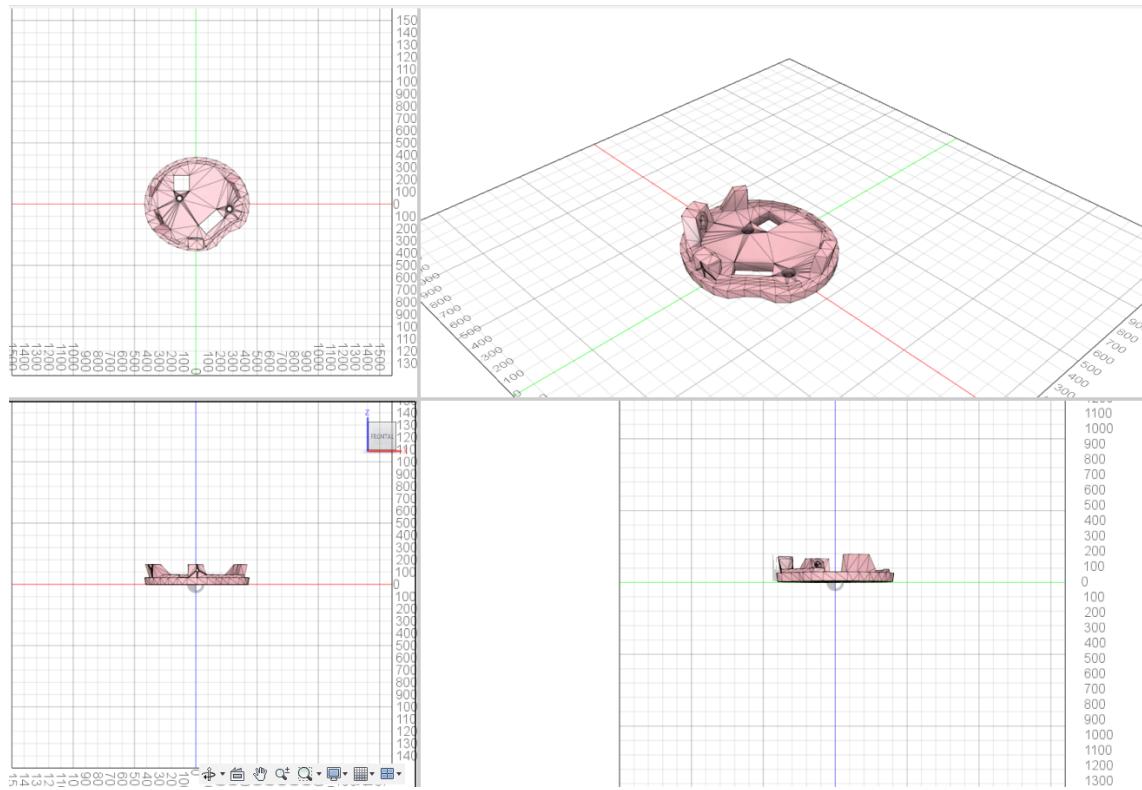


Figura 4.7: robcap3V2.stl

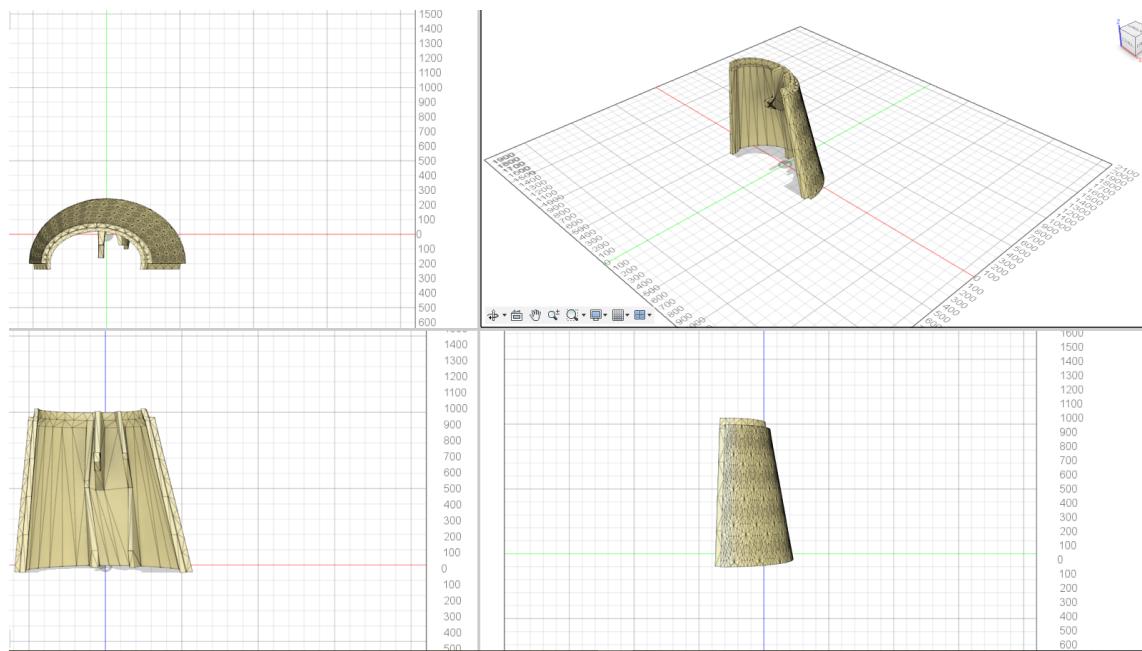


Figura 4.8: robpart2V4.stl

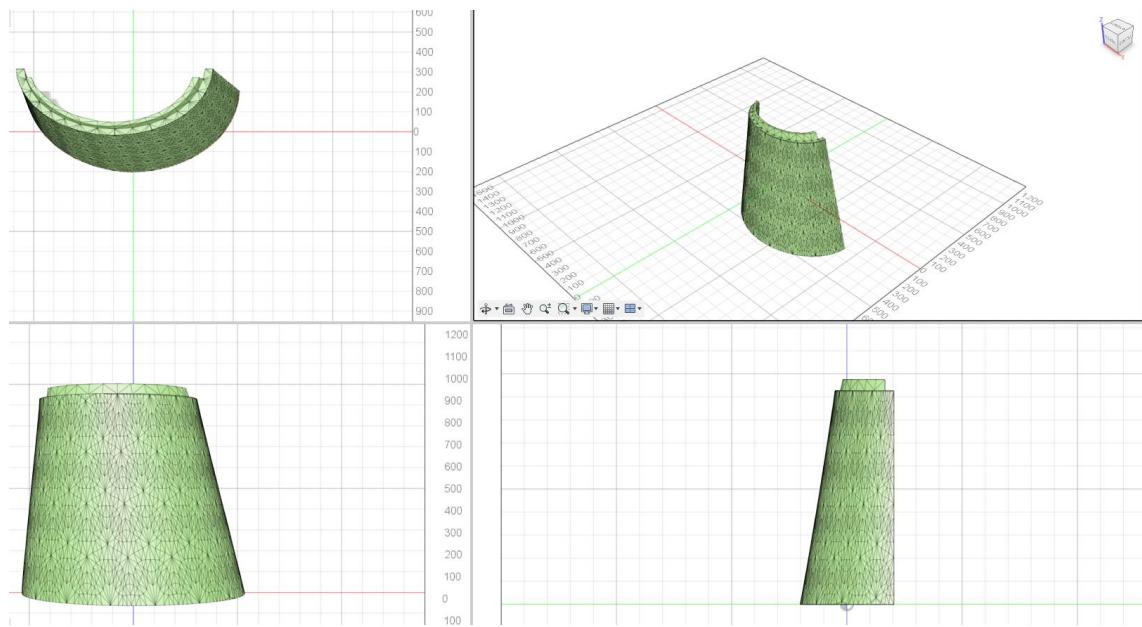


Figura 4.9: robpart3V4.stl

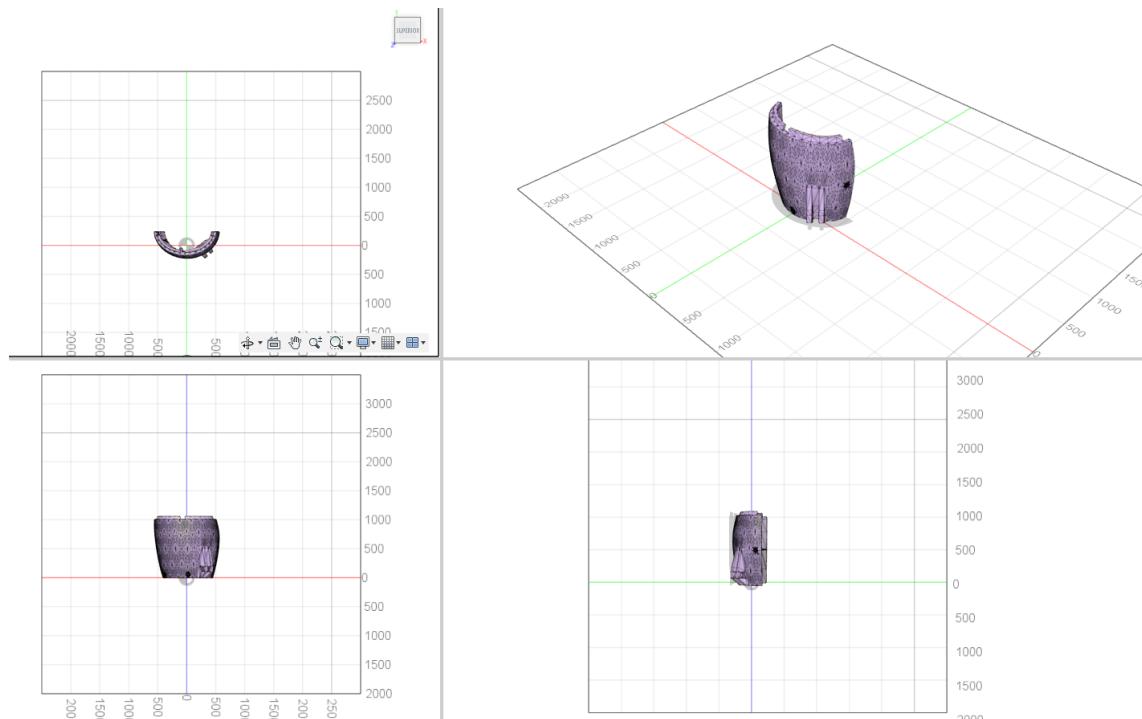


Figura 4.10: robpart4V4.stl

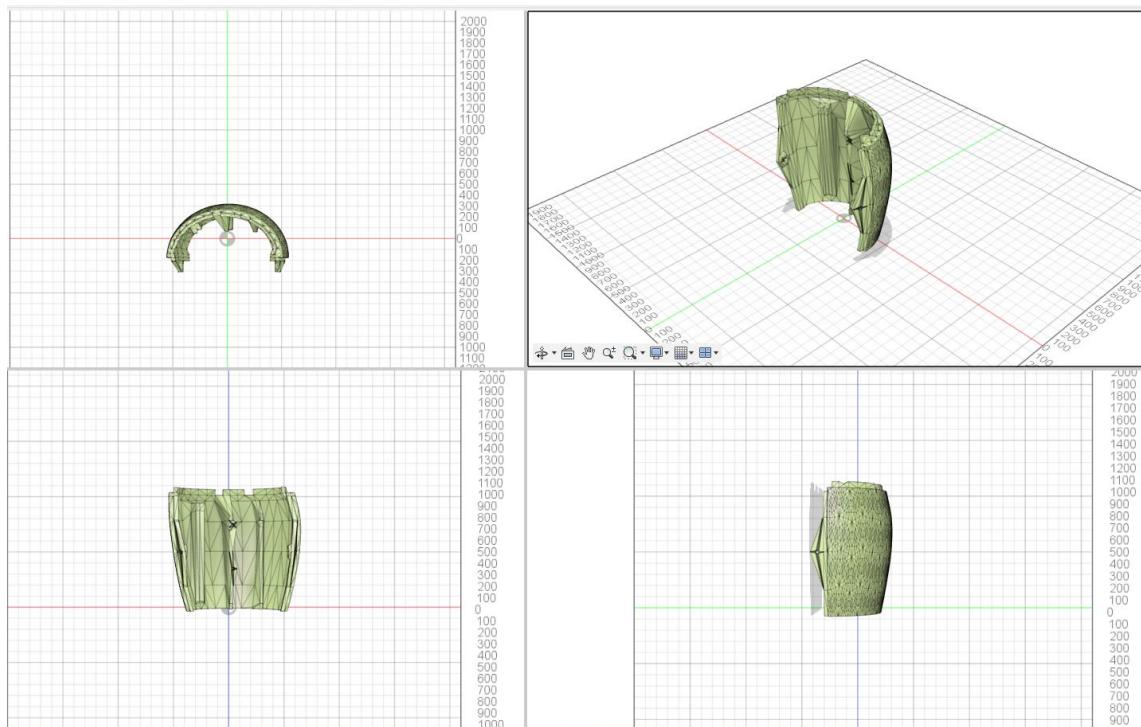


Figura 4.11: robpart5V4.stl

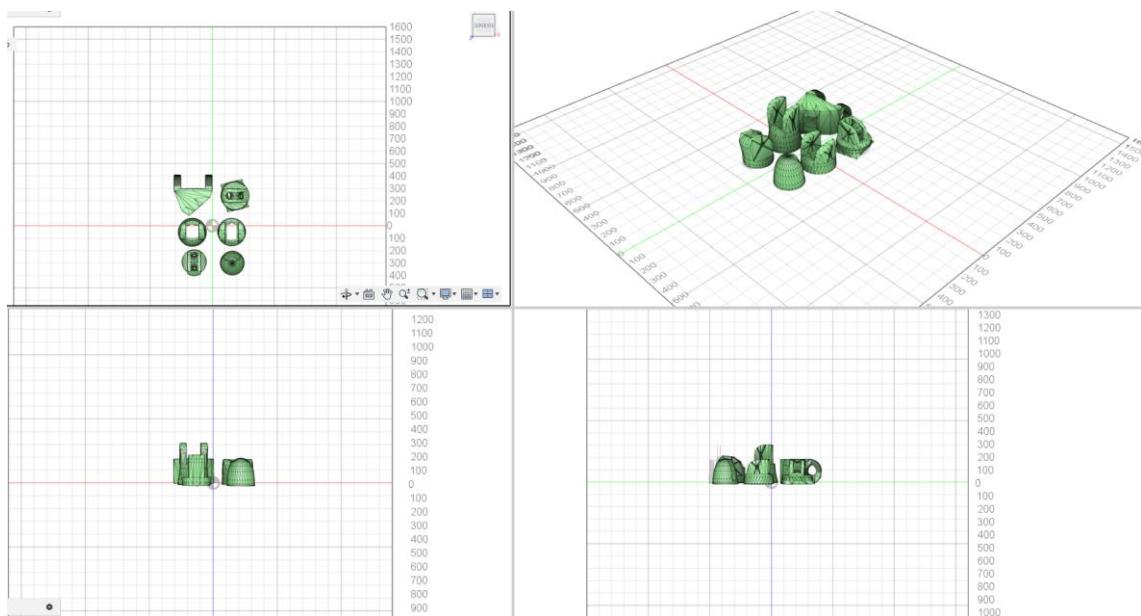


Figura 4.12: thumb5.stl

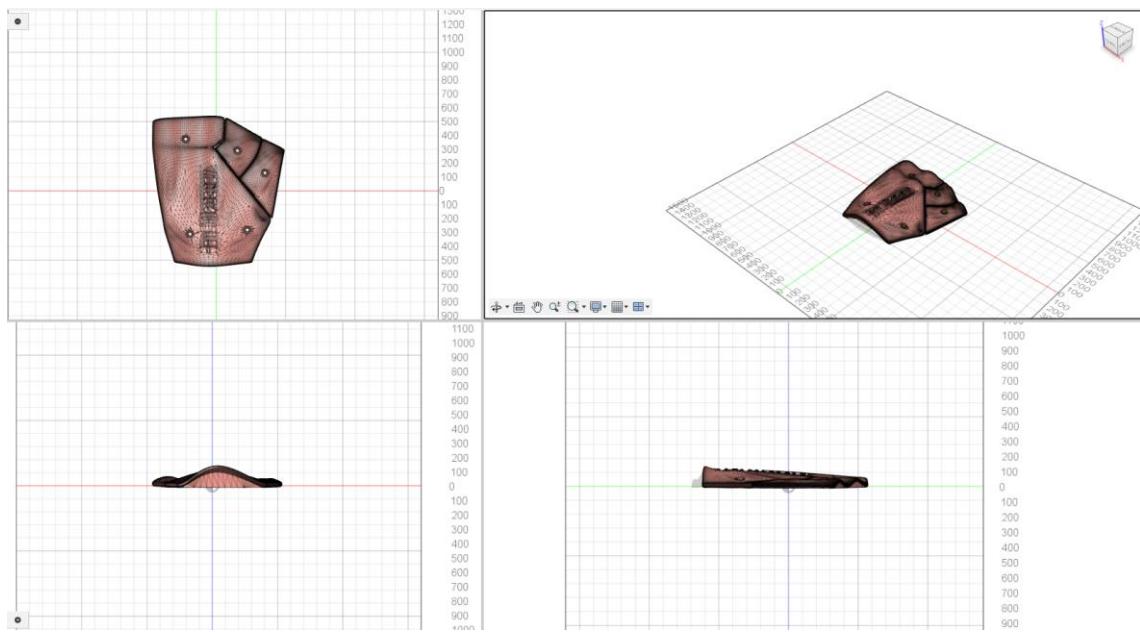


Figura 4.13: topsurface6.stl

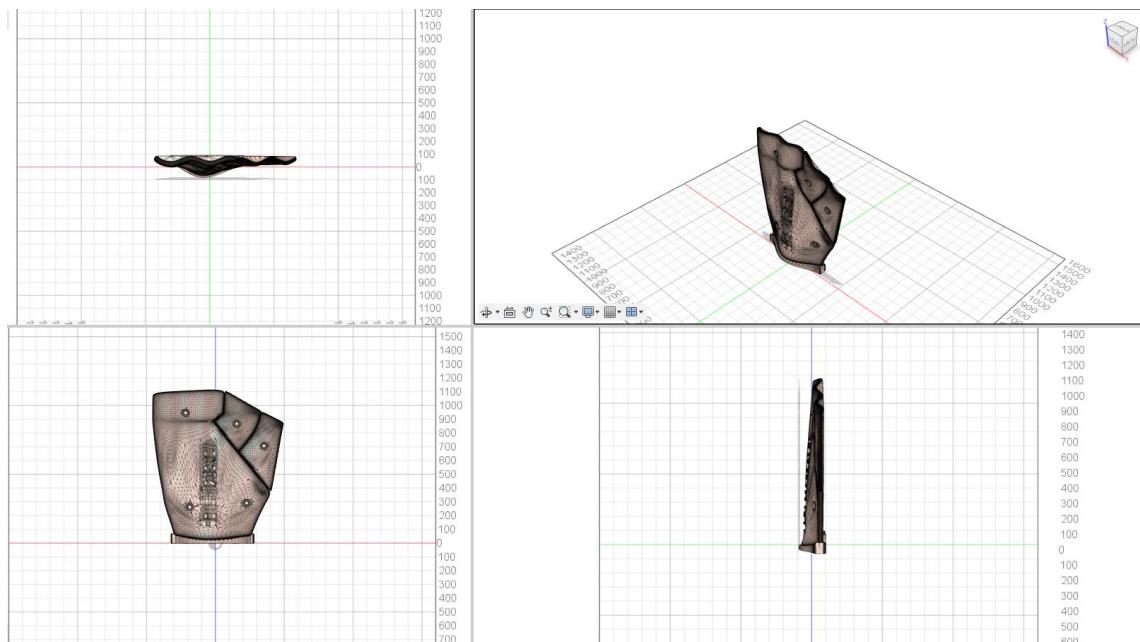


Figura 4.14: topsurfaceUP6.stl

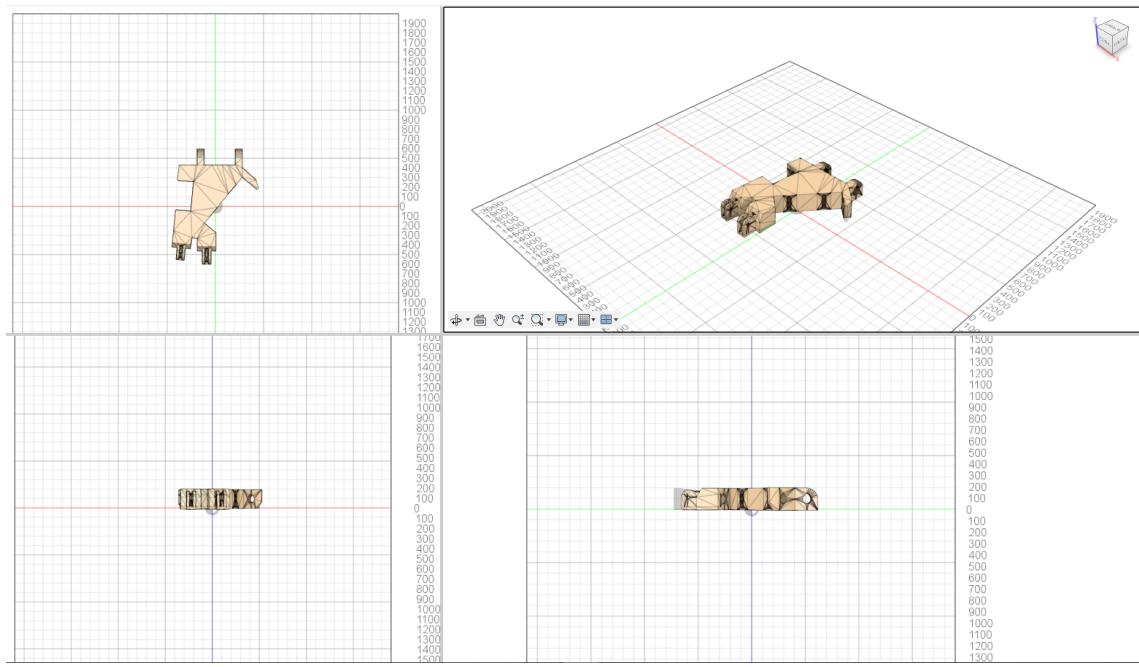


Figura 4.15: WristLargeV4.stl

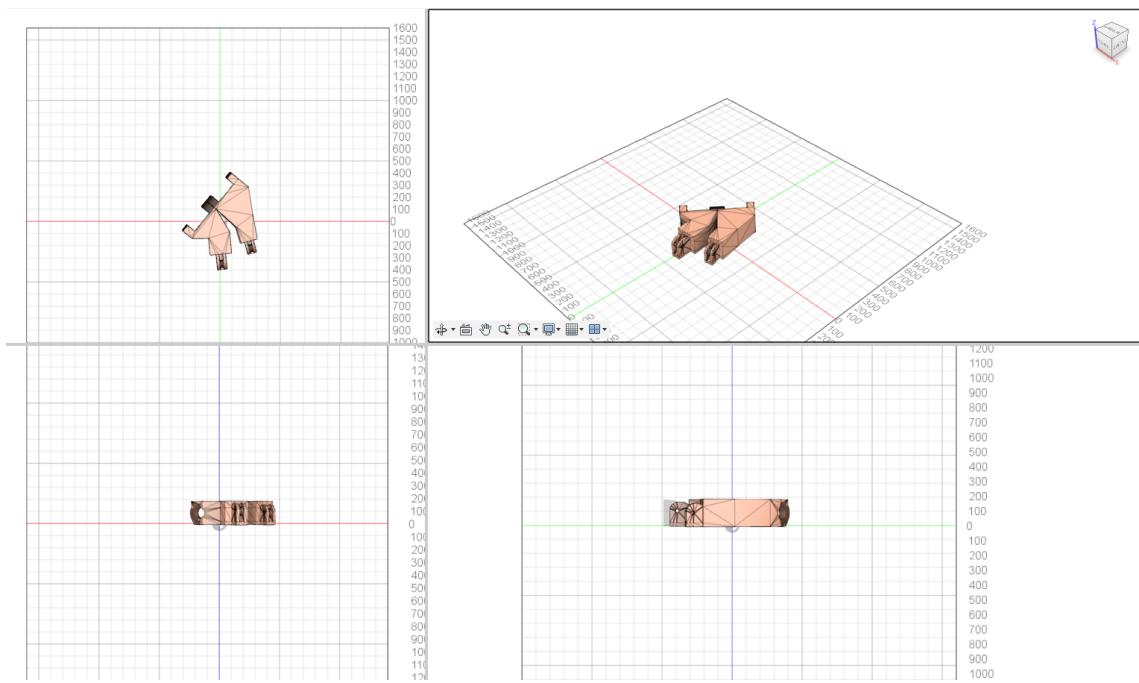


Figura 4.16: WristsmallV4.stl

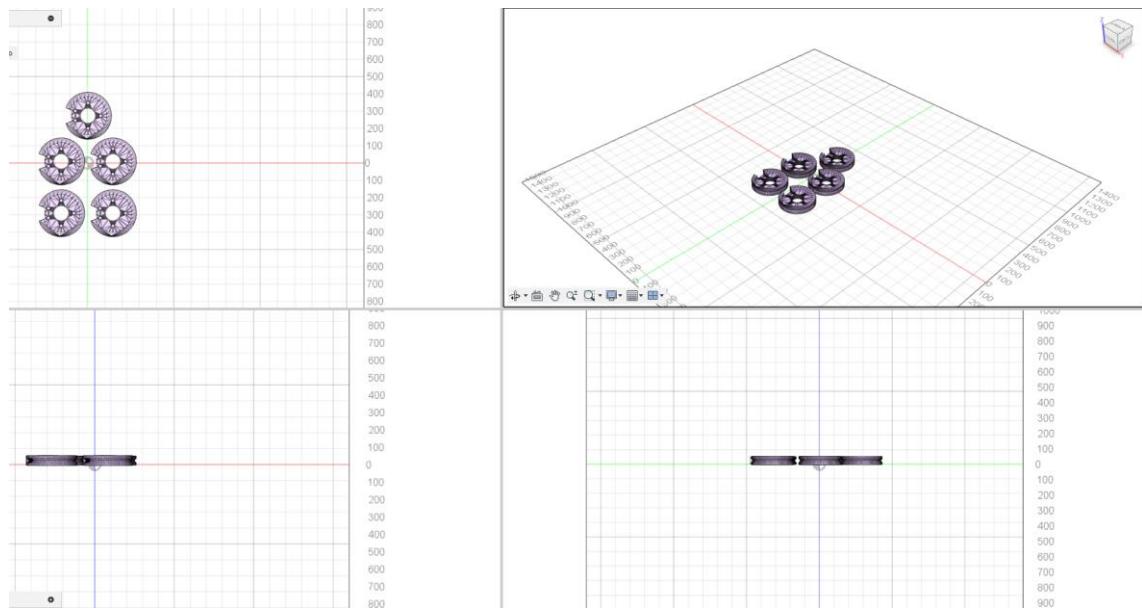


Figura 4.17: servo-pulleyX5.stl

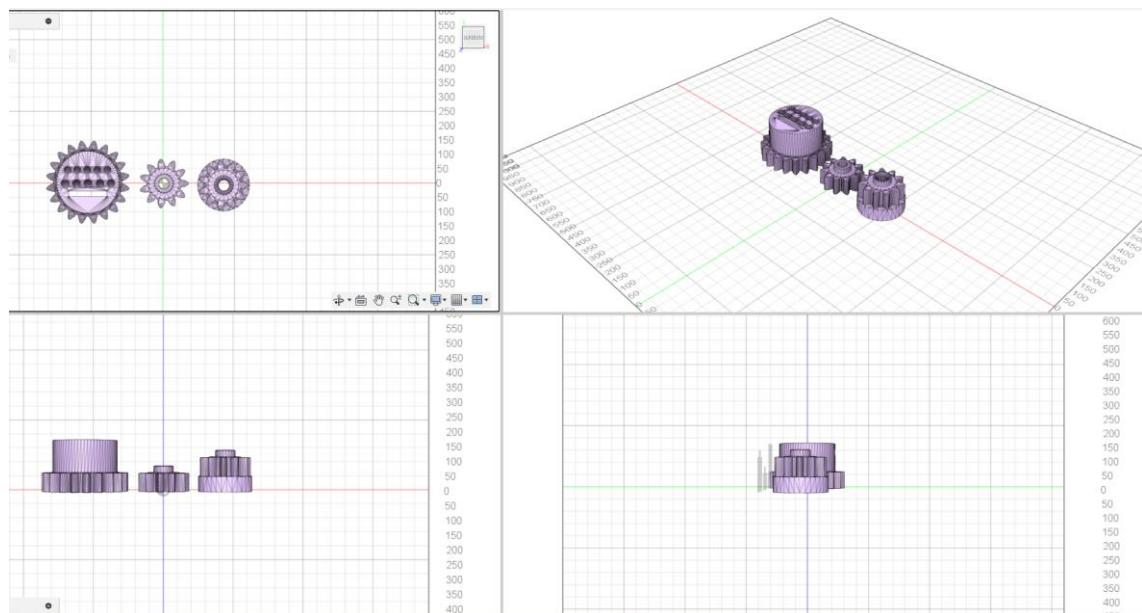


Figura 4.18: WristGearsV5.stl

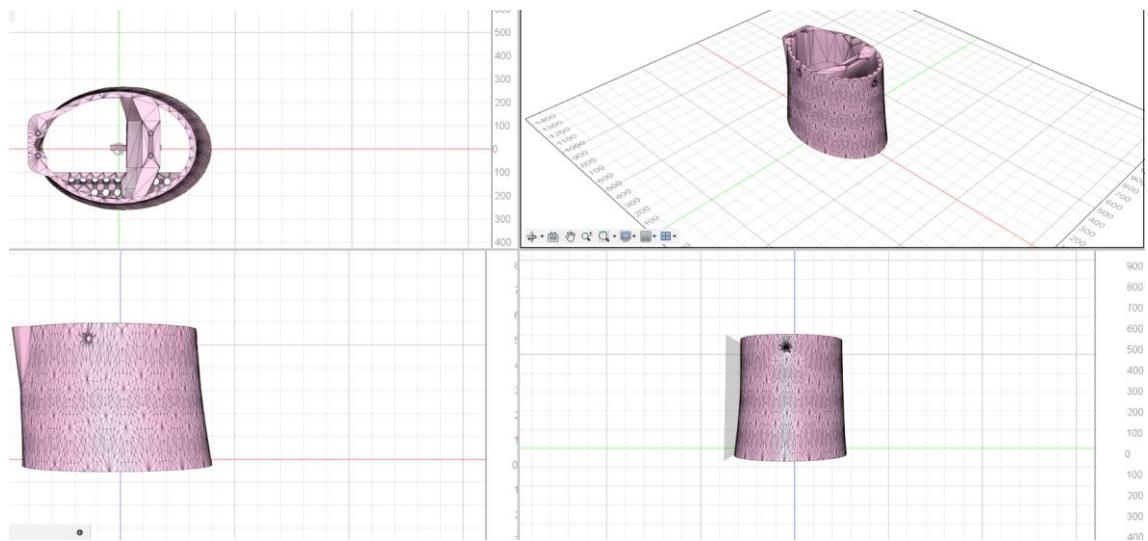


Figura 4.19: RotaWrist1V4.stl

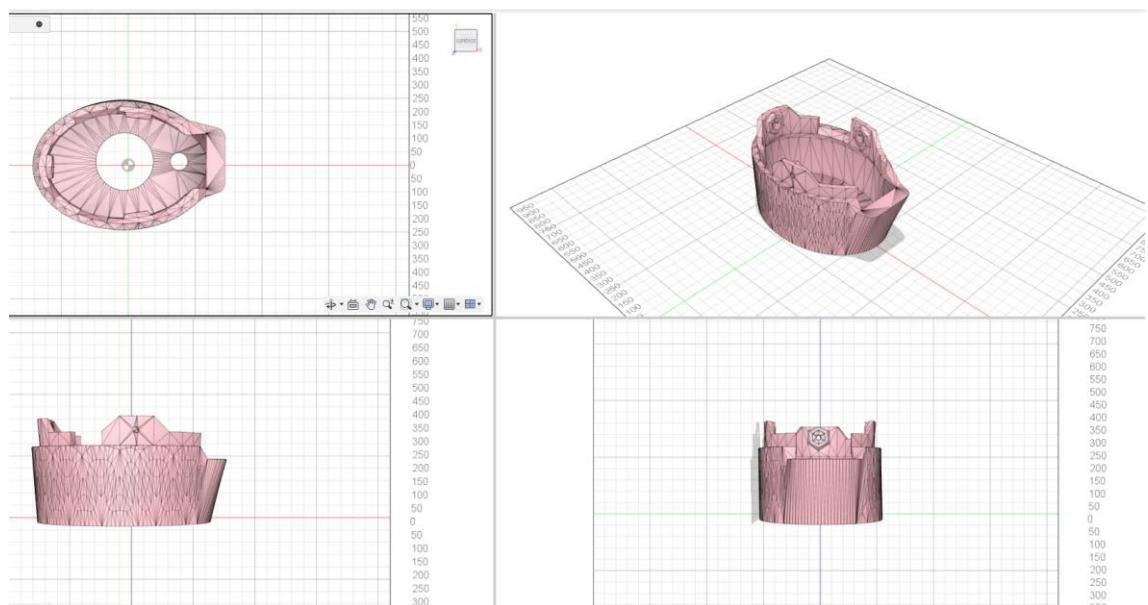


Figura 4.20: RotaWrist2V3.stl

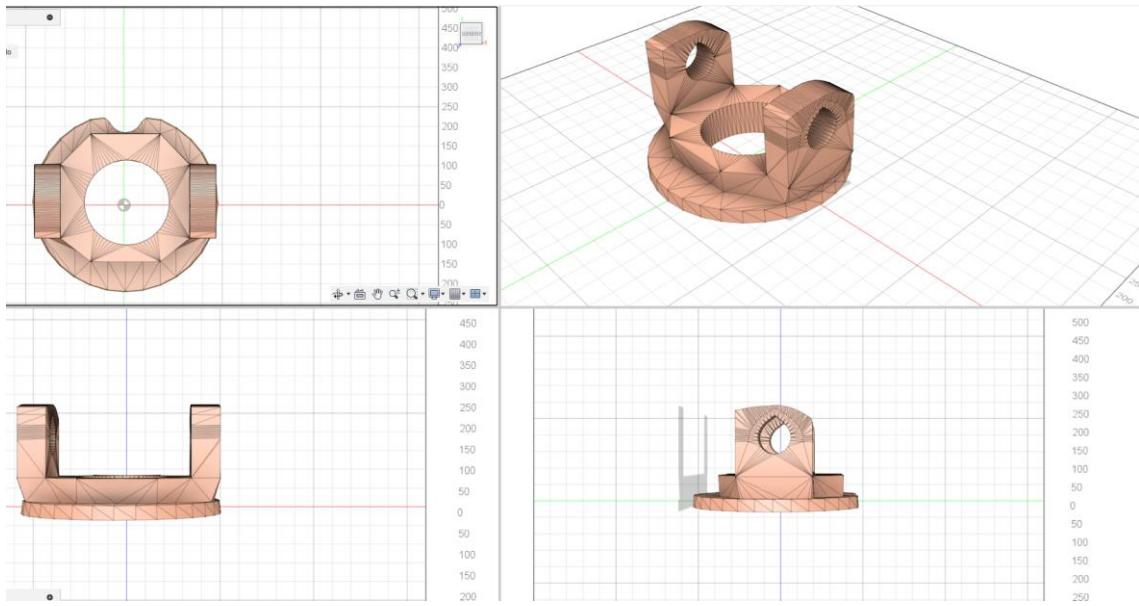


Figura 4.21: RotaWrist3V3.stl

4.2. Materiales

4.2.1. Materiales de impresión 3D

El PLA (ácido poliláctico) es un material termoplástico biodegradable derivado de recursos renovables como el almidón de maíz, lo que lo hace una opción ecológica para la impresión 3D. Es conocido por su facilidad de impresión, ya que no requiere una cama caliente y tiene una baja tendencia a deformarse. Esto lo hace ideal para crear prototipos y piezas detalladas con una alta calidad de superficie. Además, el PLA tiene una buena rigidez y una resistencia suficiente para muchas aplicaciones, aunque no es tan resistente al impacto o al calor como otros materiales como el ABS.

Para la fabricación de una mano robótica, el PLA es una opción viable debido a su precisión en la impresión y su capacidad para crear piezas complejas con detalles finos. Aunque el PLA es rígido, su resistencia puede ser adecuada para componentes de la mano que no están sujetos a cargas excesivas. Sin embargo, debido a su menor resistencia al calor y a su fragilidad relativa, es importante considerar el uso de PLA para partes estructurales secundarias o decorativas en la mano robótica, y posiblemente combinarlo con otros materiales para componentes que requieran mayor durabilidad y flexibilidad.

En el caso del proyecto, el diseño no está sometido a cargas excesivas ni a temperaturas elevadas, ya que a través de la mano no hay cables que conduzcan ningún tipo de corriente, sino cables de Nylon.

4.2.2. Materiales principales

Los motores LX16A son un tipo de servomotor con un gran esfuerzo de torsión. Esto hace que sean los motores adecuados para el proyecto elegido.

Las características principales del motor son las siguientes:

Especificaciones	Detalles
Peso	54g
Dimensiones	45.22 x 24.72 x 36.3 mm
Velocidad	0.19seg/60° a 7.4V
Precisión del servo	0.3°
Torque	17kg.cm a 6V; 19.5kg.cm a 7.4V
Corriente de bloqueo	2.4~3A
Voltaje de trabajo	6-8.4V
ID del servo	0~253
Función de retroalimentación	Soporta retroalimentación de ángulo
Longitud del cable	200mm
Rotación	0-240°
Corriente sin carga	100mA
Método de control	Comando serial UART
Velocidad de comunicación	115200
Protección	Evitar bloqueo y sobrecalentamiento
Retroalimentación de datos	Temperatura, Voltaje, Posición
Indicador	LED
Modo de trabajo	Modo servo y modo motor de deceleración
Almacenamiento	Guarda datos cuando se apaga
Engranaje	Metal

Cuadro 4.1: Especificaciones de los servomotores

Por otro lado, se ha utilizado un conversor TTL, empleando un USB2DYNAMIXEL.

El USB2DYNAMIXEL es un dispositivo empleado para controlar DYNAMIXEL directamente desde una PC. Se conecta al puerto USB de la computadora, y se incorporan conectores de 3P y 4P para facilitar la conexión de varios DYNAMIXEL. Además, el USB2DYNAMIXEL puede utilizarse para convertir el puerto USB en puerto serie en computadoras que carecen de este último, como en laptops, etc. Esta funcionalidad resulta especialmente útil cuando se conectan controladores exclusivos de DYNAMIXEL, como CM-2, CM-2+, CM-5 y CM-510, al puerto USB, o cuando se conecta ZIG2Serial al puerto USB para controlar robots de manera inalámbrica.

4.2.3. Materiales montaje

Los materiales empleados durante el montaje y ensamblado de las piezas y los motores quedan resumidos en la siguiente tabla:

Material	Nombre comercial	Cantidad	Cantidad empleada
PLA (piezas)	ELEGOO PLA 3D	-	Todas las piezas de la mano
Poliamida (Nylon)	Nylon Caperlan	500 metros	3x30cm +2x40cm = 170cm
Cinta americana	Cinta americana Gris 3M	50m	50cm
Soporte de madera	Cayro chess box	1	1 (20x13x7 cm)
Tornillos M3x10	M3x10	5	5
Tornillos LX16A	LX16A	65	10
Cinta adhesiva doble cara (Cloruro de polivinilo, Acrílico)	-	50m	20cm

Cuadro 4.2: Materiales empleados

4.3. Recursos utilizados

Para el control del hardware de este proyecto, se han utilizado varias librerías de Python disponibles en GitHub. Python es un lenguaje de programación de alto nivel, conocido por su sintaxis clara y legible, así como por su capacidad para manejar grandes cantidades de datos y aplicarles diversas funciones de manera eficiente.

Una de las librerías empleadas es OpenCV, que permite la captura de datos a través de una cámara, así como su posterior procesamiento e identificación de patrones. En este caso, se ha utilizado el módulo cvZone y su apéndice HandTrackingModule, que facilita el reconocimiento de patrones de manos mediante un algoritmo basado en inteligencia artificial. Este módulo se fundamenta en TensorFlow, una biblioteca de código abierto desarrollada por Google para el aprendizaje automático y profundo.

GitHub es una plataforma en línea que utiliza Git para el control de versiones, permitiendo a los desarrolladores almacenar, gestionar y colaborar en proyectos de código. Las librerías disponibles en esta plataforma se pueden clonar e importar en otros proyectos.

Además, se ha utilizado la librería MediaPipe, que ofrece soluciones avanzadas de procesamiento de datos en tiempo real. MediaPipe es especialmente útil para la detección y seguimiento de objetos.

La primera librería utilizada en este proyecto es una librería basada en Python que genera un panel de control para modificar todas las opciones desde un único lugar. La segunda librería incluye métodos que permiten mover los motores y leer la posición en la que se encuentra el servomotor.

4.4. Entornos desarrollo

El entorno escogido para realizar la parte de software ha sido VisualStudio Code.

Se trata de un entorno que permite elegir varios lenguajes de programación, incluir librerías y facilita la inclusión de diferentes métodos automáticamente a la hora de programar.

Por todo ello, se ha considerado el entorno óptimo donde realizar e importar las librerías especificadas previamente para la realización completa del proyecto.

Como se ha citado previamente, el proyecto está realizado en Python 3.10. en su totalidad.

5. Desarrollo

5.1. Análisis de requisitos

Los requisitos escogidos se han fundamentado en la metodología MoSCoW, la cual implementa un sistema de priorización de requisitos en función de las necesidades del sistema.

En primer lugar, se han escogido los requisitos del sistema y de usuario, y se han priorizado según la metodología anterior.

De esta forma, se resume en la siguiente tabla los requisitos principales del proyecto realizado:

MoSCoW	Requisitos
Must	Capturar los movimientos de la mano usando una cámara web con OpenCV y HandDetector de cvzone.
	Mover los servomotores según los dedos levantados detectados por el módulo HandDetector
	Implementar alguna función para controlar la posición de los servomotores según la configuración de los dedos.
	Implementar las funciones de configuración inicial de la librería que genera el panel de control y detener el movimiento de los motores.
	Permitir que el fisioterapeuta monitorice los movimientos del paciente de forma remota utilizando la mano robótica como herramienta visual.
Should	Mejorar la precisión de la detección de dedos mediante ajustes en los parámetros del detector, seleccionando las posiciones adecuadas para cada dedo.
	Añadir funcionalidad para la flexión del pulgar y la flexión interfalángica del pulgar.
	Implementar una función para movimientos específicos de los dedos índice y pulgar.
	Garantizar que la imagen capturada se volteé horizontalmente para una visualización precisa, ya que evitaría capturar la mano contraria.
Could	Integrar una interfaz de usuario para controlar y ajustar los movimientos de la mano robótica.
	Añadir funcionalidades adicionales para diferentes ejercicios de rehabilitación.
Won't	Incluir detección de manos múltiple en esta versión inicial del proyecto.
	Implementar características avanzadas de inteligencia artificial fuera del alcance actual de la rehabilitación básica.

Cuadro 5.1: Requisitos del sistema

A su vez, se han clasificado en requisitos de sistema y de usuario, para complementar lo anterior:

Requisitos de Sistema	Requisitos de Usuario
Capturar los movimientos de la mano usando una interfaz de usuario.	Poder iniciar y detener la captura de movimientos desde la interfaz de usuario.

cámara web con OpenCV y HandDetector de cvzone.	
Mover los servomotores según los dedos levantados detectados por el módulo HandDetector.	Visualizar en tiempo real los movimientos de la mano capturados por la cámara.
Implementar alguna función para controlar la posición de los servomotores según la configuración de los dedos.	Interactuar con la mano robótica a través de una interfaz de usuario intuitiva y fácil de usar.
Implementar las funciones de configuración inicial de la librería que genera el panel de control y detener el movimiento de los motores.	Monitorear los movimientos del paciente de forma remota, permitiendo al fisioterapeuta realizar ajustes según sea necesario.
Permitir que el fisioterapeuta monitoree los movimientos del paciente de forma remota utilizando la mano robótica como herramienta visual.	Poder ajustar la sensibilidad y otros parámetros de detección desde la interfaz de usuario.
Mejorar la precisión de la detección de dedos mediante ajustes en los parámetros del detector, seleccionando las posiciones adecuadas para cada dedo.	Acceder a opciones avanzadas para ejercicios específicos de rehabilitación.
Añadir funcionalidad para la flexión del pulgar y la flexión interfalángica del pulgar.	Personalizar la interfaz de usuario según las preferencias del fisioterapeuta y del paciente.
Implementar una función para movimientos específicos de los dedos índice y pulgar.	
Garantizar que la imagen capturada se voltee horizontalmente para una visualización precisa, ya que evitaría capturar la mano contraria.	

Integrar una interfaz de usuario para controlar y ajustar los movimientos de la mano robótica.	
Añadir funcionalidades adicionales para diferentes ejercicios de rehabilitación.	
Incluir detección de manos múltiple en esta versión inicial del proyecto.	
Implementar características avanzadas de inteligencia artificial fuera del alcance actual de la rehabilitación básica.	

Cuadro 5.2: Requisitos de sistema y de usuario

5.2. Desarrollo hardware

5.2.1. Desarrollo

Durante este apartado, se explicará de manera extensa el proceso realizado en cada parte del desarrollo hardware.

5.2.1.1. Desarrollo 3D y estructural

En primer lugar, se realizó una búsqueda de modelos 3D de manos fácilmente replicables, encontrándose el modelo de inMoov previamente citado.

Se procedió a imprimir todas las piezas desde una impresora Kinproon 3D 3.0, usando para todas las piezas PLA o ácido poliláctico.

Seguidamente, para la parte del antebrazo se tuvieron que realizar ciertos ajustes a la hora de montar los servomotores. El diseño inicial incluía un antebrazo y una pieza donde se atornillaban los servomotores, de manera que cupiesen todos. Sin embargo, dado que los servomotores LX16A no permitían el atornillado, y no cabían dentro del antebrazo de manera vertical, se procedió a usar un soporte de madera para soportar la estructura y facilitar la transmisión del movimiento sin entorpecerse los cables entre sí.

5.2.1.2. Desarrollo de los motores LX16A y conexión TTL

Los motores LX16A se encuentran conectados entre sí mediante un puerto que admite un cable que conduzca a tierra, otro de una entrada de voltaje y otro para la transmisión de datos.



Figura 5.1: Conexión Servo LX16A

De esta manera, se conectan en serie todos los servomotores entre sí mediante un protocolo en serie UART mediante un TTL.

TTL (Transistor-Transistor Logic) es un tipo de señal digital que se utiliza comúnmente en la electrónica para transmitir datos entre dispositivos. En el contexto de la comunicación serie UART (Universal Asynchronous Receiver/Transmitter), TTL se refiere al nivel de voltaje utilizado para representar los bits de datos. En un sistema UART, los datos se transmiten en forma de series de bits, y el nivel de voltaje TTL se utiliza para representar los bits de datos como "1" o "0".

La manera en la que se relaciona el TTL y el protocolo de serie UART radica en que el nivel de voltaje TTL se utiliza para transmitir los bits de datos a través de la comunicación serie UART. En una comunicación UART, los datos se envían en paquetes secuenciales de bits, donde el nivel de voltaje TTL se utiliza para representar la presencia de un bit "1" o "0". Esto significa que un voltaje alto podría representar un bit "1", mientras que un voltaje bajo podría representar un bit "0".

La implementación de lo anterior es necesario debido a la configuración del servomotor LX16A, el cual tiene una conversión predeterminada half-duplex.

En la conversión half-duplex los dispositivos pueden recibir y enviar datos, pero no a la vez. En el caso de querer ejecutar los servomotores es necesario un feedback, que se obtiene realizando la conversión a full-duplex, permitiendo que se reciban y se envíen datos a la vez.

Todo ello es necesario, ya que los servomotores deben mandar como feedback parámetros como la velocidad, temperatura, posición o voltaje, y sin esta conversión no sería posible.

Mediante el protocolo descrito anteriormente se consigue implementar una conversión full-duplex. Para implementar este protocolo, se ha utilizado un USB2 Dynamixel.

USB2 Dynamixel se trata de un protocolo de comunicación utilizado en los servomotores Dynamixel que se conectan a través de una interfaz USB 2.0. Estos servomotores utilizan una comunicación serie asincrónica basada en paquetes de datos para enviar comandos y recibir datos del controlador o del sistema de control. La comunicación USB2 Dynamixel es diferente de la comunicación UART en términos de protocolo y físicamente se conecta a través de un puerto USB en lugar de un puerto serie UART. Sin embargo, ambos métodos de comunicación comparten el concepto de enviar y recibir datos digitales entre dispositivos.

La diferencia entre los motores LX16A y los Dynamixel son mínimas y requieren la misma conversión para ponerse en marcha, por lo que el uso de este protocolo permite realizar la conversión que se busca.

Por otro lado, se necesita también una fuente que suministre el voltaje necesario para que los servomotores LX16A funcionen correctamente, en concreto de 12V.

Por tanto, el esquema a seguir de la conexión de los servomotores al ordenador seguiría el siguiente:

Entrada del primer servomotor conectada a la fuente de alimentación, la fuente de alimentación conectada sucesivamente con el USB2 Dynamixel configurado como TTL, y este último iría como entrada USB al ordenador.

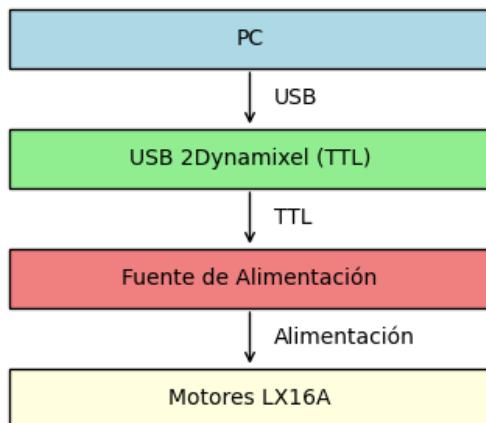


Figura 5.2: Diagrama de conexión del hardware del sistema

Para concluir, se descartó la idea de usar un Arduino en la conversión TTL por falta de material. Para realizar la conversión en los motores LX16A hacía falta usar un integrado concreto, el 74HC126N, que es el que junto con una resistencia de 10k Ohmios permitía la conversión a full-dúplex. Sin embargo, este integrado no se ha encontrado disponible ni en ninguna tienda que pudiese mandarlo en menos de 15-20 días.

Por otro lado, se tenía el integrado 74LS2471N, el cual sirve para los motores de la marca Dynamixel, unos servomotores con características muy similares a los LX16A.

Aún teniendo el conexionado de referencia de este integrador, no se logró realizar el conexionado de manera correcta, ya que se tenía como referencia cómo hacerlo con el integrado 74HC126N para los motores LX16A y el 74LS2471N para los Dynamixel. Teniendo el integrado 74LS2471N y los motores LX16A no se logró realizar el conexionado completo.

Por tanto, se optó por la opción de la conversión con el USB2 Dynamixel y realizar el proyecto sin Arduino.

5.2.2. Montaje y ensamblado

El montaje de las piezas ha consistido en diferentes etapas que pueden quedar esquematizadas de la siguiente manera:

5.2.2.1. Impresión 3D de las piezas

Durante esta etapa, se ha realizado la impresión de las piezas de la mano con filamentos de PLA en diferentes tandas.

En la siguiente tabla se resumen los tiempos por pieza y los tiempos de entrega:

Nombre del archivo de la pieza (en formato .stl)	Tiempo de impresión	Correspondencia
Arduinosupport	2 días en total	Soporte para el Arduino
Auriculaire3		Dedo meñique
Bolt_entretoise7		Tubos de soporte entre varias piezas
Majeure3		Dedo corazón o medio
Robpart2V4*	11 horas	Antebrazo derecho
Robcap3V2*	6 horas	Soporte inferior del brazo
Ringfinger3	4 horas	Dedo anular
Robpart3V4*	13 horas	Antebrazo derecho
Coverfinger1*	2 horas	Piezas estéticas para los dedos
Robpart4V4*	13 horas	Soporte del antebrazo derecho
Robparts5V4*	13 horas	Soporte del antebrazo derecho
Thumb5	4 horas	Dedo pulgar
WristsmallV4	8 horas	Pieza secundaria de la palma de la mano
Topsurface6	5 horas	Soporte para la mano de carácter estético
WristLargeV4	9 horas	Pieza principal de la palma de la mano
topsurfaceUP6	6 horas	Soporte para la mano de carácter estético
Index3	4 horas	Dedo índice
Servo-pulleyX5	4 días en total	Ruedas para los motores
WristGearsV5*		Engranajes para la muñeca
RotaWrist3V3		Pieza pequeña de la muñeca
RotaWrist1V4		Pieza intermedia de la muñeca
RotaWrist2V3		Pieza grande de la muñeca

Cuadro 5.3: Tiempos de impresión 3D

Todas las piezas marcadas con un asterisco (*) no se han incluido finalmente en el proyecto.

Las piezas robpart2V4, robpart3V4, robpart4V4 y robpart5V4 forman el antebrazo. Posteriormente, se comprobó que para incluir los motores dentro del antebrazo que conforman estas piezas se debía incluir una pieza que actuase como soporte de los servomotores y se atornillase al brazo, la cual no serviría para los motores LX16A, ya que el diseño original estaba orientado a otro tipo de servomotores con posibilidad de ser atornillados y de menor tamaño.

La pieza robcap3V2 tampoco se ha incluido en el diseño final al ser el soporte que cierra la estructura del antebrazo que conforman las piezas anteriores.

También, las piezas coverfinger1, topsurface6 y topsurfaceUP6 no se han incluido ya que son de carácter estético y no estructural en el diseño, por lo que se han descartado.

Por último, los engranajes que tiene la pieza WristGearsV5 no se han incluido tampoco, ya que se decidió descartar que la muñeca tuviese capacidad de movimiento.

5.2.2.2. Preparación de las piezas de la mano

Durante esta fase se han preparado todas las piezas para su posterior ensamblado, ya que algunas de ellas al imprimirse habían adquirido cierto relieve que impedía su unión.

En primer lugar, se ha procedido a limar todas las piezas que forman los dedos, ya que la mayoría de ellas eran demasiado gruesas para ensamblarse y debían perder algo de volumen. Estas serían las piezas Auriculaire3.stl, Majeure3.stl, Thumb5.stl, Index3.stl y ringfinger3.stl.

Por otro lado, se ha tenido que limar el tubo de Bolt_entretoise.stl, ya que era demasiado grueso para que cupiese por la pieza wristLargeV4.stl.

También, se ha hecho uso de un taladro con una broca de ancho de 6mm para perforar los agujeros de la pieza wristsmallV4.stl y permitir que los tubos de la pieza anterior cupiesen.

5.2.2.3. Montaje de las piezas de la mano

Durante esta etapa se ha montado la estructura de la mano completa.

En primer lugar, se han pegado las piezas de los dedos con un pegamento especial para plásticos. En concreto, en las 5 piezas que forman los dedos se incluyen las falanges por separado, por lo que cada pieza realmente tiene a su vez 5 piezas que forman un dedo.



Figura 5.3: Dedo completo montado

Por tanto, se han pegado las falanges medias y las proximales por separado, encajando posteriormente las falanges distales y la media con la proximal para formar el dedo completo.

Todo el proceso anterior se debe a que los dedos deben actuar como una articulación, por lo que se deben pegar algunas piezas para que se mantenga la estructura, pero a la vez esté articulado. Es por ello por lo que se encajan las falanges entre sí, provocando la rotación de estas articulaciones.

Por otro lado, se ha utilizado la pieza wristsmallV4.stl para encajar los dedos anular y meñique, sosteniéndose la estructura a su vez con uno de los tubos proporcionados en la pieza Bolt_entretoise7.stl. También se ha encajado el dedo pulgar con otro de los tubos de la pieza anterior, ya que el movimiento del pulgar debe ser acompañado con un eje auxiliar, no como el resto de los dedos.

Una vez se han acoplado todas las piezas anteriores, se ha terminado de proporcionar una estructura fija a los dedos. Para ello, se deben ligar las articulaciones para evitar que se caigan las piezas que estén enlazadas y no pegadas, y seguir proporcionando el giro necesario.

Para ello, se han empleado distintos materiales dependiendo del tamaño de cada falange y de cada dedo.

Para las falanges distales de los dedos corazón, anular y meñique se ha utilizado una cuerda elástica blanca de poliamida (Nylon) de 1mm de grosor. Se ha empleado este tipo ya que son los agujeros más pequeños de todas las falanges y la elasticidad de la cuerda permitía su encaje. Sin embargo, esto ha sido un inconveniente a la hora de flexionar y extender los dedos, ya que la elasticidad de la cuerda permitía tener cierta holgura en la articulación e impedía el movimiento. Para tener un movimiento óptimo la holgura en la articulación debe ser prácticamente nula.

Para las falanges distales del dedo índice y pulgar se ha empleado una cuerda gris más rígida de poliéster de 0,5mm de grosor. Al ser más rígida, aunque menos ancha, limitaba la holgura para estos dedos de una manera más eficiente que la anterior.

La cuerda anterior se ha usado también para las falanges medias del pulgar y del meñique por el mismo motivo.

Para los otros tres dedos se han empleado cables rígidos de 1mm de grosor de PVC. Estos cables eran más eficientes para estas falanges medias, ya que el material que los recubre es más rígido y evita que haya holgura en estas falanges. Además, su grosor es mayor, por lo que se une a la rigidez para evitar una mayor holgura.

Por último, para las falanges proximales se ha utilizado un cable de 1mm con recubrimiento de PVC para todos menos para el dedo corazón, donde se ha empleado uno de un diámetro de 1,5mm. Todo ello ha proporcionado que las falanges estén fijas y no se muevan entre sí, dando un efecto real de pieza compacta.

Una vez la mano estaba montada, se procedió a unirla a un soporte, el cual dan las piezas rotawrist1V4.stl, rotawrist2V3.stl y rotawrist3V3.stl. Entre sí se unen de tal manera:

La pieza rotawrist3V3.stl se enlazó con la pieza wristLargeV4.stl mediante uno de los tubos de Bolt_entretoise7.stl y a su vez con las piezas rotawrist2V3.stl y rotawrist1V4 rotawrist3V3.stl para terminar de conformar la estructura.

Con todo ello, se obtuvo la estructura final de la mano:



Figura 5.4: Extensión de la mano completa

5.2.2.4. Estudio de la transmisión del movimiento

En este apartado, se describirá la parte de cómo se mueven los dedos con los motores y el proceso que se ha seguido para llegar a ello.

En primer lugar, se descartó el uso de cables eléctricos para mover los dedos, ya que se tendría que incluir un circuito eléctrico más complejo para mover todos los dedos. Por tanto, se seleccionó como material un hilo de pesca Caperlan de 500m de longitud, y 18,7kg de resistencia a la tracción.

Características del Nylon:

El **nylon** es una poliamida lineal termoplástica sintética, lo que significa que es una molécula grande cuyos componentes están unidos por enlaces específicos. Fue producido por primera vez en 1935 por el químico estadounidense Wallace Carothers en las instalaciones de investigación de DuPont en Delaware. Carothers creó lo que se conoce como nylon 66, una de las variantes más comunes en la actualidad.

El nylon tiene una variedad de aplicaciones debido a sus características excepcionales. Es fuerte, resistente a la abrasión y a la absorción de humedad, duradero, resistente a los productos químicos, elástico y fácil de lavar. Estas propiedades lo hacen adecuado para reemplazar metales de baja resistencia en muchos contextos.

Una de las principales aplicaciones del nylon es en la fabricación de ropa y como refuerzo en materiales de caucho como neumáticos de coche. También se utiliza en la fabricación de cuerdas o hilos y en piezas moldeadas por inyección para vehículos y equipos mecánicos.

El nylon posee una alta resistencia a la tracción y una notable estabilidad dimensional, lo que lo hace ideal para usos que requieren durabilidad y resistencia a la degradación. Tiene un alto punto de fusión, lo que le permite resistir el calor y la fricción, y es menos susceptible a la decoloración una vez teñido.

Debido a estas características, el nylon es ampliamente utilizado en la vida cotidiana y en diversas industrias. Por ejemplo, se utiliza para hacer paracaídas debido a su ligereza y resistencia al agua, y en trajes de baño por su naturaleza impermeable. También es común en la fabricación de piezas de máquinas, accesorios del hogar como alfombras y cuerdas, y muchos otros productos industriales.

El nylon reciclado tiene beneficios ambientales significativos, desviando residuos de los vertederos y utilizando menos recursos en su producción comparado con el nylon virgen. Gran parte del nylon reciclado proviene de redes de pesca viejas, lo que ayuda a reducir la basura en los océanos.

Debido a sus propiedades de alta resistencia a la tracción, durabilidad, resistencia a la abrasión y estabilidad dimensional, el nylon es un material excelente para la fabricación de hilos de pescar. Los hilos de pescar de nylon son capaces de soportar el peso y la lucha de los peces grandes sin romperse fácilmente. Su elasticidad ayuda a absorber los tirones bruscos de los peces, reduciendo el riesgo de rotura. Además, su resistencia a los productos químicos y al agua lo hace ideal para su uso prolongado en ambientes acuáticos.

Los hilos de pescar de nylon también son relativamente fáciles de manejar y enrollar, y su resistencia al desgaste asegura una larga vida útil incluso en condiciones difíciles.

Desarrollo:

Por todo ello, se escogió como el material óptimo para resistir los esfuerzos de flexión y extensión que moverían los dedos.

Se procedió a coger cables de Nylon de 50cm de longitud y se colocaron por dentro de los dedos. La estructura de las piezas está diseñada originalmente para ello, por lo que hay dos cables por dedo. Uno de ellos se coloca por la parte posterior del dedo y otro por la anterior, de manera que no se crucen entre sí al tensionarse y permitan una transmisión del movimiento óptima.

Una vez se han colocado los cables, se deben pasar por los agujeros de la pieza rotawristV3.stl para evitar que se enrollen entre sí. De esta manera, se consigue que los cables pasen por todos los dedos permitiendo su flexión y extensión y evitando que se enrollen entre sí. Por último, se realizan varios nudos al final de los dedos para evitar que al realizar cualquier esfuerzo se salgan de los dedos.

De esta manera, sólo falta enlazar estos cables con algo que provoque estos esfuerzos.

Para ello, se han utilizado los motores LX16A.

Como se comentaba previamente, las piezas robpart2V4.stl, robpart3V4.stl, robpart4V4.stl, robpart5V4.stl y robcap3V2.stl no se incluyen en el proyecto final. Esto se debe a que a la hora de colocar los motores verticalmente dentro del antebrazo que forman estas piezas, los cables de Nylon se enrollan entre sí. El diseño original tiene una pieza que permite acoplar los motores ahí y atornillarlos a la estructura, haciendo que la transmisión del movimiento sea posible.

Sin embargo, los motores LX16A son mayores que los servomotores del diseño original, y además no permiten ser atornillados a la pieza. Por todo ello, se descartó incluirlos en el diseño original del antebrazo y se tuvo que buscar una alternativa.

Para transmitir el movimiento, se debía encontrar la forma de poder tirar de un cable o de otro dependiendo de si se buscaba la extensión o la flexión del dedo. Para ello, se imprimieron las piezas servopulleys5.stl, las cuales son unas poleas que permitían enrollar los cables de Nylon en ella.

La alternativa a ello hubiera sido pegar directamente los cables a la rueda de los motores, pero sería mucho más frágil y propenso a que se soltase cualquier cuerda. Además, al mover alguna de las cuerdas cuando se gire hacia un sentido u otro, la otra cuerda donde no se esté provocando el esfuerzo se queda más distendida y menos tirante, por lo que si se pegan las cuerdas no se produciría este hecho y no se podría mover en ningún sentido el motor para mover los dedos.

Por todo ello, se escogió colocar estas poleas en los motores para enrollarlas y así tener cierto margen a la hora de tirar de alguna de las cuerdas.

Para realizar esto, primero se observó cómo colocar estas poleas para que estuviesen fijadas al eje del motor y de esa forma no deslizasen. Esto es muy importante para la

transmisión del movimiento, ya que, si las ruedas no están fijadas al eje del motor LX16A y deslizan, no se transmite el movimiento a los dedos.

El motor tiene un eje central donde se acopla la polea. Además, estos motores incluyen unas ruedas de diferentes diámetros que se acoplan al eje del motor.

Por tanto, se escogió la rueda de 2cm de diámetro del motor y se colocó dentro de la polea, la cual tiene unos 3cm de diámetro y un hueco central de más de 2cm para poder colocar esta rueda.

La idea es unir la polea y la rueda mediante un tornillo al eje central del motor, de manera que las piezas queden fijas al eje y se pueda enrollar las cuerdas en la polea. La rueda quedaría conectada al eje y a la polea para conectar ambas cosas.

Sin embargo, el motor tiene un relieve en la parte inferior que sobresale, que evita que la polea quepa si se coloca junto a la rueda, ya que chocaría con la parte inferior y se doblaría.

Para evitar esto, se colocó una arandela de 1,5cm de diámetro en la parte posterior de la rueda, de manera que la rueda sobresalga en el eje y no llegue al final, provocando que la polea choque con la parte inferior del motor. Se coloca un tornillo desde la rueda hasta el eje uniendo la rueda dentro de la polea al eje finalmente.

Una vez se tiene la estructura de la rueda y polea sujetas al eje se deben enrollar los cables en la polea y fijar tensos en la estructura. Para ello se colocan dos tornillos de 0,5cm de longitud y 1mm de diámetro en los huecos de la rueda.

De esta manera, se consigue fijar los cables de Nylon una vez estén tensos en la polea, presionándolos con los tornillos y unas arandelas de 0,5cm de diámetro. Al atornillar estos tornillos con las arandelas sobre los cables se quedan fijos a la rueda.

Por último, se debe colocar una arandela para el tornillo central de 0,5cm de diámetro (un poco menor a la del tornillo), de forma que no se hunda, ya que el diámetro del tornillo es menor que el de la rueda, por lo que no aguantaría fijo.



Figura 5.5: Motor con la polea incorporada

5.2.2.5. Colocación de los motores

En esta fase se explica detalladamente la colocación de los motores para el correcto funcionamiento del sistema.

Para comenzar, se ha utilizado una caja de madera que previamente se utilizaba como caja de piezas de ajedrez para soportar los motores. Esta caja se eligió dado que tiene la posibilidad de colocar los motores en dos plantas y es más ancha que el antebrazo descartado. También, es resistente a cualquier tipo de esfuerzo y aguanta el peso de la mano sometida a esfuerzos de manera eficiente.

Tras descartar el antebrazo original, se pensó en utilizar una caja de corcho y colocar los cinco motores en fila. Sin embargo, esto tiene un problema principal, que provoca que los cables se enrollen con el eje si el motor está alejado del eje vertical que forman la mano y los motores.



Figura 5.6: Mano alineada con eje vertical

Por tanto, colocar los 5 motores en línea no sería posible ya que como se ha explicado en apartados anteriores la rueda sobresale junto a la polea para no chocar con la parte inferior del motor, creando un hueco entre la polea y el motor.

Para corregirlo, se buscó un soporte que pudiese colocar a todos los motores en el eje vertical que forman la mano y los motores.

En la parte superior de la caja se realizaron 12 agujeros con un taladro eléctrico y una broca de 3mm de diámetro, aprovechando finalmente 10 de ellos para poder pasar todos los cables.

También, se realizaron 6 agujeros del mismo diámetro en el soporte central para poder pasar los cables hacia los motores de la parte inferior de la caja. Finalmente, se aprovecharon 4 de ellos.

Se colocaron los cables de Nylon en los motores según la proximidad a los agujeros y motores y se enrollaron en las poleas tal y como se ha descrito previamente.

Finalmente, para fijar los motores se ha empleado cinta aislante de doble cara con resistencia hasta [x kilos], para que aguante la tensión el motor cuando se mueva.

También, se han conectado entre sí los motores para el conexionado en serie de los servomotores LX16A, de manera que quedan de la siguiente forma:



Figura 5.7: Disposición de los motores

Los números de identificación de los motores están seleccionados para que correspondan con los parámetros del software, pero principalmente para que se correspondan con el dedo que mueven.

Dedo	Número de identificación del motor (id)
Pulgar	1
Índice	2
Corazón o medio	3
Anular	4
Meñique	5

Cuadro 5.4: ID de los motores

5.2.3. Resultado

El resultado final es una estructura compuesta por la mano impresa y completa, con todos los hilos de Nylon conectados a los motores de la caja donde están colocados y pegados.

A su vez, el motor que corresponde al pulgar está conectado a la fuente de 12V externa, que a su vez está conectada al USB2 Dynamixel para la conversión de half-duplex a full-duplex.

Todo ello quedaría resumido en el siguiente esquema:

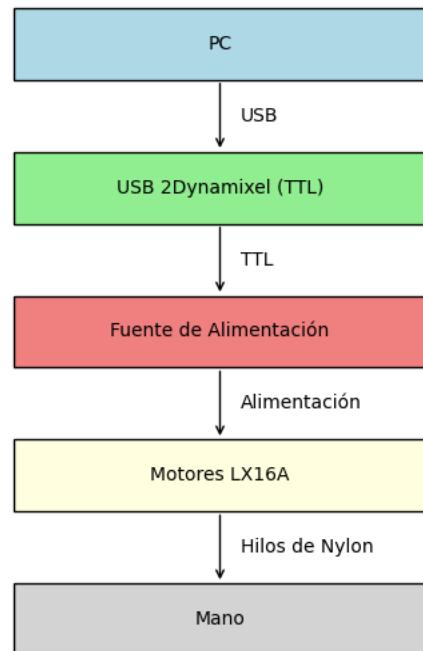


Figura 5.8: Diagrama completo del sistema

5.3. Desarrollo software

Durante este apartado se resumirá el desarrollo del software utilizado en el proyecto y se explicará su aplicación y relación con el hardware.

5.3.1. Desarrollo inicial del código

5.3.1.1. Búsqueda de librerías LX16A

En este apartado se busca explicar las ideas iniciales del proyecto para realizar los movimientos de rehabilitación que se buscan en la mano.

En primer lugar, se hizo una búsqueda sobre algunas librerías escritas en el lenguaje Python que sirviesen para los motores LX16A. Se encontraron varias, pero se eligieron dos principalmente.

La primera, nos proporciona una librería con todas las funciones aplicables a los motores LX16A que nos permite el hardware. Incluye leer datos de sensores de temperatura o voltaje, feedback de posición, velocidad o número de identificación, además de poder modificar estos tres últimos parámetros.

Con ella, se realizó el código completo para mover los motores.

La segunda se utilizó para la configuración manual de los motores y para la lectura de datos. La librería incluye un GUI (Graphical User Interface) desde el cual se pueden detectar todos los motores conectados y se pueden extraer y modificar valores de los parámetros disponibles.

Un **GUI** (Graphical User Interface) es una interfaz gráfica de usuario que permite interactuar con una computadora o un dispositivo a través de elementos gráficos y visuales como botones, iconos, ventanas, menús y otros componentes interactivos. A diferencia de las interfaces de línea de comandos (CLI), donde los usuarios deben escribir comandos de texto, las GUIs ofrecen una forma más intuitiva y visual de interactuar con el sistema.

Características de un GUI:

Elementos gráficos: Los GUIs utilizan componentes visuales como ventanas, botones, menús, iconos y cuadros de diálogo.

Interacción con el usuario: Permiten la interacción mediante dispositivos de entrada como el ratón, el teclado y pantallas táctiles.

Facilidad de uso: Su diseño intuitivo permite a los usuarios realizar tareas sin necesidad de conocer comandos complejos.

Retroalimentación visual: Proporcionan respuestas visuales inmediatas a las acciones del usuario, mejorando la experiencia y eficiencia.

Historia y Evolución:

Los primeros GUIs fueron desarrollados en la década de 1970 en el Xerox Palo Alto Research Center (PARC). El Xerox Alto fue uno de los primeros sistemas en implementar una interfaz gráfica. Apple popularizó los GUIs con el lanzamiento del Apple Macintosh en 1984, que introdujo conceptos como el escritorio, ventanas, iconos y menús desplegables. Microsoft siguió con su propio GUI en Windows.

Componentes Comunes

Ventanas: Áreas de trabajo que pueden contener diferentes elementos y que se pueden mover y redimensionar.

Botones: Elementos que pueden ser clicados para ejecutar una acción.

Menús: Listas desplegables que ofrecen opciones al usuario.

Iconos: Representaciones gráficas de programas, archivos o funciones.

Cuadros de diálogo: Ventanas emergentes que solicitan la entrada del usuario o proporcionan información.

Aplicaciones:

Los GUIs son omnipresentes en sistemas operativos modernos (Windows, macOS, Linux), aplicaciones de software (Microsoft Office, Adobe Creative Suite), y dispositivos móviles (iOS, Android).

5.3.1.2. Búsqueda de recursos para la captura de gestos

Para este apartado, se describe en detalle la búsqueda de herramientas para capturar los gestos de una mano. El objetivo principal es que se detecte la mano mediante una cámara y se almacenen los datos dependiendo de los movimientos realizados.

Para ello, se ha decidido usar la librería OpenCV integrada en el lenguaje Python 3.10 junto a la librería Mediapipe de Google.

La librería **OpenCV** se encarga de la captura de imágenes principalmente, mientras que la librería de Mediapipe detecta concretamente las manos y procesa una serie de datos en función de lo que se capture.

OpenCV (Open Source Computer Vision Library) es una biblioteca de software libre especializada en visión por computadora y aprendizaje automático. Fue desarrollada originalmente por Intel y está escrita en C y C++, con interfaces en varios lenguajes como Python, Java y MATLAB. OpenCV se ha convertido en una de las bibliotecas más populares y utilizadas para aplicaciones de procesamiento de imágenes y visión por computadora debido a su amplia funcionalidad y facilidad de uso.

Características Principales de OpenCV:

1. Procesamiento de imágenes:

Filtrado de imágenes: Aplicación de filtros para suavizado, detección de bordes, etc.

Transformaciones geométricas: Rotación, escalado, recorte y transformación de perspectiva.

Corrección de color: Conversión entre diferentes espacios de color, ajuste de brillo y contraste.

2. Detección y reconocimiento de objetos:

Detección de características: Uso de detectores como SIFT, SURF, ORB.

Reconocimiento de rostros: Clasificadores en cascada basados en Haar y modelos de redes neuronales.

Seguimiento de objetos: Algoritmos para el seguimiento de objetos en video, como el seguimiento de puntos y el seguimiento basado en contornos.

3. Visión 3D:

Reconstrucción 3D: Uso de técnicas de estéreo visión para reconstruir escenas en tres dimensiones.

Calibración de cámaras: Herramientas para calibrar cámaras y corregir distorsiones de lente.

4. Machine Learning:

Algoritmos integrados: Implementaciones de algoritmos de aprendizaje supervisado y no supervisado, como SVM, KNN, árboles de decisión, y más.

Integración con bibliotecas de aprendizaje profundo: Soporte para modelos entrenados en TensorFlow, PyTorch y Caffe.

5. Interoperabilidad y extensibilidad:

Multiplataforma: Compatible con sistemas operativos como Windows, Linux, macOS, Android e iOS.

Interfaces en múltiples lenguajes: Compatible con C++, Python, Java, y MATLAB, lo que permite a los desarrolladores trabajar en el entorno de su elección.

Aplicaciones Comunes de OpenCV:

Seguridad y Vigilancia: Sistemas de detección y reconocimiento de intrusos.

Automoción: Sistemas de asistencia al conductor (ADAS), como la detección de carriles y obstáculos.

Salud: Análisis de imágenes médicas para diagnóstico y tratamiento.

Robótica: Navegación autónoma y manipulación de objetos.

Realidad Aumentada: Superposición de información digital en el mundo real.

Por otro lado, la librería Mediapipe una biblioteca de código abierto desarrollada por Google que facilita la creación y despliegue de soluciones de visión por computadora y aprendizaje automático en tiempo real. La biblioteca está diseñada para ser eficiente y fácil de usar, permitiendo a los desarrolladores integrar rápidamente capacidades avanzadas en sus aplicaciones.

Entre sus características principales, Mediapipe es multiplataforma, soportando Android, iOS y sistemas de escritorio como Linux, macOS y Windows. Esto permite a los desarrolladores implementar soluciones en una amplia variedad de dispositivos. La biblioteca es modular, compuesta por módulos reutilizables llamados calculadores, que realizan tareas específicas como la detección de rostros, la estimación de poses y el seguimiento de manos. Estos módulos pueden ser combinados y personalizados para crear pipelines complejos. Además, Mediapipe está optimizado para tiempo real, diseñado para ejecutar eficientemente en dispositivos móviles y otros sistemas con recursos limitados, garantizando un rendimiento en tiempo real.

Mediapipe incluye una variedad de soluciones preentrenadas para tareas comunes como detección y seguimiento de manos, estimación de poses y seguimiento del cuerpo, reconocimiento facial y seguimiento de la malla facial, detección de objetos y seguimiento del iris. Estas soluciones preentrenadas facilitan la implementación de capacidades avanzadas sin necesidad de entrenar modelos desde cero.

Los usos comunes de Mediapipe incluyen interacción humano-computadora mediante gestos, aplicaciones de realidad aumentada, seguimiento de actividad física y rehabilitación, análisis de expresiones faciales y emociones, así como aplicaciones en el ámbito de la salud y el bienestar. La biblioteca es una herramienta poderosa para desarrolladores interesados en integrar visión por computadora y aprendizaje automático en sus aplicaciones de manera eficiente y efectiva.

Con todo ello, se han combinado ambas librerías para extraer los datos capturados.

5.3.1.3. Configuración de los motores mediante GUI

En esta fase, se ha utilizado el GUI de la librería LX16A comentada anteriormente para configurar el número de identificación o ID de los motores.

La configuración de los ID de los motores ha sido necesaria mediante esta librería externa ya que por defecto los motores venían todos con el mismo ID y al ejecutar el método. `readID()` de la otra librería aparecía un error por consola.

Al indagar un poco en ello, se comprobó que mediante el GUI conectando uno a uno los motores (y no en serie) se podía ver que el ID de los motores era siempre el mismo, por lo que al ejecutar cualquier función saltaba un error.

Por tanto, se procedió a conectar uno a uno los motores al TTL y a cambiar mediante el GUI los ID, enumerándolos del 1 al 5 en orden.

Por otro lado, el GUI nos permite identificar la posición de los motores, por lo que por defecto se colocaron todos los motores en la posición 500 en el rango de [0,1000]. De esta manera, se pueden colocar los cables de Nylon tensos a los motores ya colocados en la misma posición, y se pueden ajustar mejor los movimientos.

También, el GUI nos permite ver si salta algún tipo de aviso por sobrecalentamiento en los motores, si se ha bloqueado el rotor o hay una sobrecarga en el motor por exceso de voltaje.

5.3.2. Desarrollo completo del código

5.3.2.1. Importación

En este apartado se explicará el código incluido paso a paso.

En primer lugar, se deben importar las librerías anteriormente mencionadas:

```

1 import cv2
2 from cvzone.HandTrackingModule import HandDetector
3 from LX16A import LX16A

```

Código 5.1: Importación librerías

La librería cv2 se encarga de la captura de imágenes, mientras que los módulos de cvzone de HandTrackingModule y HandDetector se encargarán de la detección de puntos de referencia de la mano. Por tanto se importan la librería openCV y Mediapipe para esta parte.

cvzone es una biblioteca de Python que simplifica la implementación de varios algoritmos de visión por computadora utilizando OpenCV y Mediapipe. Una de sus características destacadas es el módulo HandTrackingModule, que facilita la detección y seguimiento de manos utilizando la tecnología de Mediapipe.

Por último, se importa también la librería LX16A para el uso de los motores, además de la clase LX16A, que incluye todas las funciones de la librería.

5.3.2.2. Primer ejercicio de flexión extensión completa

```

6. motor =LX16A()
7.
8. # Función para mover los motores según los dedos levantados
9. def flexion_extension(fingers):
10.    positions = [(200, 900), (100, 600), (200, 600), (50,
      500), (0, 500)]
11.    for i in range(5):
12.        speed = 1000 if i == 0 else 800
13.        if fingers[i] == 1:
14.            motor.moveServo(i+1, positions[i][0], speed)
15.            if motor.readPosition(i+1) == positions[i][0]:
16.                motor.moveServoStop(i+1)
17.            elif fingers[i] == 0:
18.                motor.moveServo(i+1, positions[i][1], speed)
19.                if motor.readPosition(i+1) == positions[i][0]:
20.                    motor.moveServoStop(i+1)
21.            else:
22.                motor.moveServoStop(i+1)
23.

```

Código 5.2: Ejercicio flexión y extensión

En primer lugar, se importa la clase LX16A, la cual se asigna a la variable **motor**. De esta forma, se importa la clase completa y se itera siempre sobre ella.

1. La función **flexión_extension** recibe como parámetro la lista de 5 dígitos de **fingers**. Posteriormente se explicará con más detalle.
2. La lista de tuplas **positions** contiene las posiciones a las que se deben mover los motores para realizar la flexión y extensión completa de los dedos. Los motores tienen un rango de 0-1000 en cuanto a las posiciones. Las posiciones

seleccionadas han sido probadas para cada motor mediante el GUI de la librería externa y fijadas a 500, de manera que las posiciones de la función **flexión_extension** han sido seleccionadas en base a la posición inicial de 500 de todos los motores.

3. Se selecciona un rango de 0-4, ya que hay 5 dígitos en la lista **fingers**; y al emplear range me selecciona de 0 hasta n-1, en este caso 4.
4. Se define la variable **speed** con dos valores. Si el valor del iterable es igual a 0 la velocidad será 1000, y si el valor es distinto a 0 será 800. Se ha implementado de esta manera ya que la velocidad necesaria para bajar los dedos se decidió que fuese más alta a la de subirlos, correspondiéndose con los valores seleccionados.
5. La parte iterativa del código se divide en dos partes:
 - a. Si el valor del dígito de la lista **fingers** es igual a 1 se implementa el método **moveServo** de la librería importada, moviendo los motores 1-5 a las posiciones de la lista de tuplas, correspondiéndose con el primer valor de las tuplas (el valor inferior). Se mueven con una velocidad **speed** que depende de la condición anterior.
 - i. Para comprobar que los motores se paren al llegar a la posición esperada, se implementa el método **readPosition**. Si la posición de la tupla seleccionada coincide con la del motor, éste se para (mediante el método **moveServoStop**).
 - b. Si el valor del dígito de la lista **fingers** es igual a 0 se implementa el método **moveServo** de la librería importada, moviendo los motores 1-5 a las posiciones de la lista de tuplas, correspondiéndose con el segundo valor de las tuplas (el valor superior). Se mueven con una velocidad **speed** que depende de la condición anterior.
 - i. Para comprobar que los motores se paren al llegar a la posición esperada, se implementa el método **readPosition**. Si la posición de la tupla seleccionada coincide con la del motor, éste se para (mediante el método **moveServoStop**).
 - c. En caso de que el valor de **fingers** no sea ni 0 ni 1 se paran todos los motores. Esta condición se ha puesto por seguridad, ya que la lista de **fingers** por defecto siempre tendrá valores de 0 y 1. Sin embargo, se han encontrado errores de que los motores seguían funcionando a pesar de no estar detectando nuevos valores de **fingers**, por lo que para evitar posibles complicaciones se ha incluido esta condición.

5.3.2.3. Segundo ejercicio de flexión del pulgar:

```
4.     def flexion_pulgar(fingers):  
5.         if fingers[0] == 1:  
6.             motor.moveServo(1, 200, 200)  
7.         else:  
8.             motor.moveServo(1, 900, 200)
```

Código 5.3: Ejercicio flexión del pulgar

La función **flexión_pulgar** es una adaptación de la anterior para únicamente el índice 0, que se relaciona con el motor con id =1, que es el que se encarga de mover el dedo pulgar. Por tanto, se han seleccionado las posiciones del ejercicio anterior con velocidad 200 para realizar este ejercicio.

5.3.2.4. Tercer ejercicio de flexión interfalángica del pulgar:

```
6.     def flexion_interfalangica_pulgar(fingers):
7.         if fingers[0] == 1:
8.             motor.moveServo(1, 300, 600)
9.         else:
10.            motor.moveServo(1,800,700)
```

Código 5.4: Ejercicio flexión interfalángica del pulgar

El ejercicio es similar al anterior, pero con la diferencia de que las posiciones y la velocidad varían menos.

El rango de movimiento partiendo de la posición inicial de 500 es de 300-800, por lo que se reduce respecto al rango del movimiento de flexión extensión completa.

También, se reduce la velocidad a 600 y 700 en lugar de 200. La velocidad en esta librería sigue un orden inverso, cuanto mayor sea el valor menor será la velocidad. Realmente, esto se explica ya que el parámetro corresponde a **rate** realmente, el cual sigue un orden inverso.

De esta forma, se consigue que únicamente se muevan la falange distal y la media y se realice este ejercicio.

5.3.2.5. Cuarto ejercicio de pinza índice:

```
4.     def pinza_indice(fingers):
5.         positions = [(200, 900), (100, 600)]
6.         for i in range(2):
7.             if fingers[i]==1:
8.                 motor.moveServo(i+1,positions[1][0],300)
9.             elif fingers[i]==0:
10.                motor.moveServo(i+1,positions[0][1],300)
```

Código 5.5: Ejercicio pinza del índice

En este ejercicio se escogen los dos primeros dedos únicamente, el pulgar y el índice, para realizar los movimientos. La función es similar a la primera pero adaptada a dos dedos. La velocidad se escoge como 300 por defecto, ya que se ha comprobado que es con la que mejor funcionan los dedos para llegar a las posiciones especificadas.

5.3.2.6. Función de restablecer posición:

```
6.     def set_motores():
7.         for i in range(0,5):
8.             motor.moveServo(i+1,500,200)
9.             if motor.readPosition(i+1)== 500:
10.                 motor.moveServoStop(i+1)
```

Código 5.6: Función restablecer posición de motores

La función **set_motores** se ha creado para restablecer las posiciones a 500 cada vez que se cierre la ejecución del programa. De esta manera, los motores parten siempre desde la misma posición y evita tener que cambiar las posiciones cada vez que se ejecute el programa en función de dónde hayan acabado al ejecutarse previamente las funciones.

Posteriormente, se hace una llamada a la función cada vez que se termine de ejecutar el programa para que todos los motores vuelvan a la posición original.

5.3.2.7. Captura de imagen y creación de lista de valores:

```
4.     cap = cv2.VideoCapture(1)
5.     detector = HandDetector(maxHands=1, detectionCon=0.9)
6.
7.     while True:
8.         success, img = cap.read()
9.         img = cv2.flip(img, 1)
10.        hands, img = detector.findHands(img, flipType=False)
11.
12.        if hands:
13.            hand = hands[0]
14.            fingers = detector.fingersUp(hand)
15.            flexion_extension(fingers)
16.            #flexion_pulgar(fingers)
17.            #flexion_interfalangica_pulgar(fingers)
18.            #pinza_indice(fingers)
19.
20.
21.            cv2.imshow("Image", img)
22.            if cv2.waitKey(1) == ord('o'):
23.                break
24.
25.    cap.release()
26.    cv2.destroyAllWindows()
27.    set_motores()
```

Código 5.7: Captura de imagen

1. **cap = cv2.VideoCapture(0):** Inicia la captura de video desde la cámara. El parámetro 1 indica que se utilizará la cámara predeterminada del sistema. En caso de que no haya más cámaras conectadas se debe seleccionar el 0 para la

predeterminada del sistema. Para conectarla a cámaras externas se deben usar los valores de 0 en adelante, exceptuando el 1.

2. **detector = HandDetector(maxHands=1, detectionCon=0.9):** Crea una instancia del detector de manos de cvzone con la configuración de detectar un máximo de una mano (**maxHands=1**) y con una confianza de detección del 90% (**detectionCon=0.9**). Lo normal es sobre el 70-100% para que detecte correctamente las referencias de la mano, de lo contrario puede dar fallos en la detección y tomar otros objetos como manos.
3. **while True:** Inicia un bucle infinito para procesar los frames de la cámara en tiempo real. Se incluye todo el código posterior dentro de este bucle.
4. **success, img = cap.read():** Lee un frame de la cámara. **success** es un booleano que indica si la lectura se realizó correctamente, y **img** contiene la imagen capturada.
5. **img = cv2.flip(img, 1):** Invierte la imagen horizontalmente para que los movimientos sean más intuitivos (como mirarse en un espejo). Se ha seleccionado este modo ya que por defecto
6. **hands, img = detector.findHands(img, flipType=False):** Detecta las manos en la imagen usando el detector de manos. **hands** contiene la información de las manos detectadas y **img** es la imagen con anotaciones (como dibujos de las manos y puntos clave).
7. **if hands::** Comprueba si se han detectado manos. Equivalente a si se detectan es True.
8. **hand = hands[0]:** Toma la primera mano detectada de la lista de manos. En este caso se especifica aunque sólo se detecte una mano en realidad.
9. **fingers = detector.fingersUp(hand):** Obtiene el estado de los dedos (levantados o bajados) de la mano detectada. Se genera una lista de 5 dígitos que servirá como parámetro para las funciones creadas para los movimientos.
10. **flexion_extension(fingers):** Llama a la función **flexion_extension** para mover los servos de los dedos según el estado de los dedos (levantados o bajados).

11. Las siguientes líneas están comentadas, ya que solo se puede ejecutar una de las funciones a la vez para evitar interferencias entre ellas. Por tanto, cuando se quiera ejecutar otra de estas funciones se debe comentar la anterior que está en ejecución y quitar el comentado de la que se quiera ejecutar.

#flexion_pulgar(fingers): Para controlar la flexión del pulgar.

#flexion_interfalangica_pulgar(fingers): Para controlar la flexión interfalángica del pulgar.

#pinza_indice(fingers): Para controlar la pinza entre el pulgar y el índice.

12. cv2.imshow("Image", img): Muestra la imagen procesada (con anotaciones) en una ventana llamada "Image". El título se puede cambiar, pero se ha elegido dejar "Image" para que sea algo general.

13. if cv2.waitKey(1) == ord('o'): Espera 1 ms a una tecla, y si se presiona la tecla 'o', rompe el bucle, terminando así el proceso de captura y procesamiento.

Esto sirve para terminar la ejecución de cualquiera de las funciones al cerrar la cámara. De esta forma, se incluye un interruptor en la captura de imágenes.

14. cap.release(): Libera la cámara, deteniendo la captura de video.

15. cv2.destroyAllWindows(): Cierra todas las ventanas creadas por OpenCV.

16. set_motores(): Llama a la función set_motores para restablecer a los servos a la posición inicial de 500. Esta función se ejecuta siempre que se detenga la ejecución del programa.

5.3.3. Resultados esperados

Los resultados esperados son los siguientes por orden:

1. Aparece en consola la siguiente línea:

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Código 5.8: Consola

La línea "INFO: Created TensorFlow Lite XNNPACK delegate for CPU." es un mensaje informativo que indica que se ha creado un delegado de TensorFlow Lite XNNPACK para la CPU.

Este mensaje se genera como parte de la inicialización del entorno de ejecución para TensorFlow Lite. El delegado XNNPACK es una opción de optimización que permite acelerar la inferencia de modelos de aprendizaje automático en dispositivos con CPU mediante el uso de operaciones aceleradas de bajo nivel.

En rasgos generales, sirve para mejorar el uso de los algoritmos basados en IA implementados con las librerías importadas de captura y procesamiento de imágenes.

2. A la vez que aparece el mensaje anterior en consola, se crea la ventana de la imagen en función de la cámara seleccionada.
3. Al poner cualquier mano sobre la cámara se debe detectar, especificar si la mano es la derecha o la izquierda, y mover los motores dependiendo de la función que se haya ejecutado.

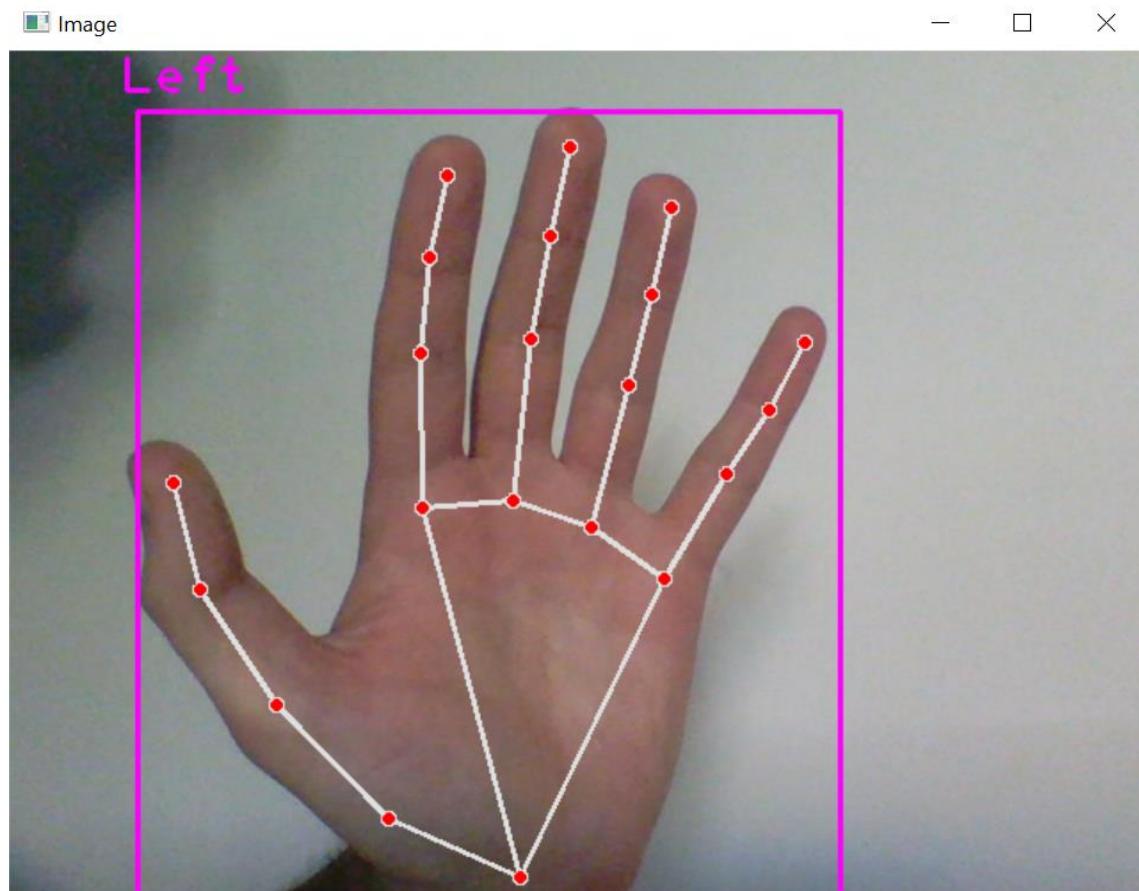


Figura 5.9: Detección de la mano

4. Internamente, aunque no se esté mostrando, se está generando una lista (**fingers**) unos y ceros dependiendo de la posición de los dedos. En función de esos valores se irán moviendo los servos.

6. Cinemática del sistema

6.1. Introducción del sistema

En este apartado se resume todo lo relacionado con las posiciones que toma la mano en los diferentes ejercicios que realiza. En apartados anteriores, se especifican las posiciones que debe tomar el motor para mover los dedos a una posición u otra, pero en este apartado se tendrá en cuenta la posición respecto a varios ejes de coordenadas para determinarlas.

Para comenzar, se debe entender primero la anatomía de un dedo para poder subdividir los ejes de referencia posteriormente:

(No aparece la imagen debido a los derechos de autor. Se recomienda visualizar la imagen: **Figure 1. Anatomic Structure of the finger with the flexor (green and yellow lines) and extensor tendons (red line)** del artículo [1] de las referencias)

En esta imagen se observan las diferentes articulaciones del dedo en función de las falanges, y los tendones que sirven de apoyo estructural para esas articulaciones y permiten los movimientos del dedo.

En condiciones ideales, un dedo humano posee 4 GDL (Grados de Libertad), 3 corresponden a la flexión-extensión de cada una de las falanges y otro a la abducción y aducción del dedo completo.

Una vez se ha descrito la anatomía del dedo completa, se procede a simplificarlo de la siguiente manera:

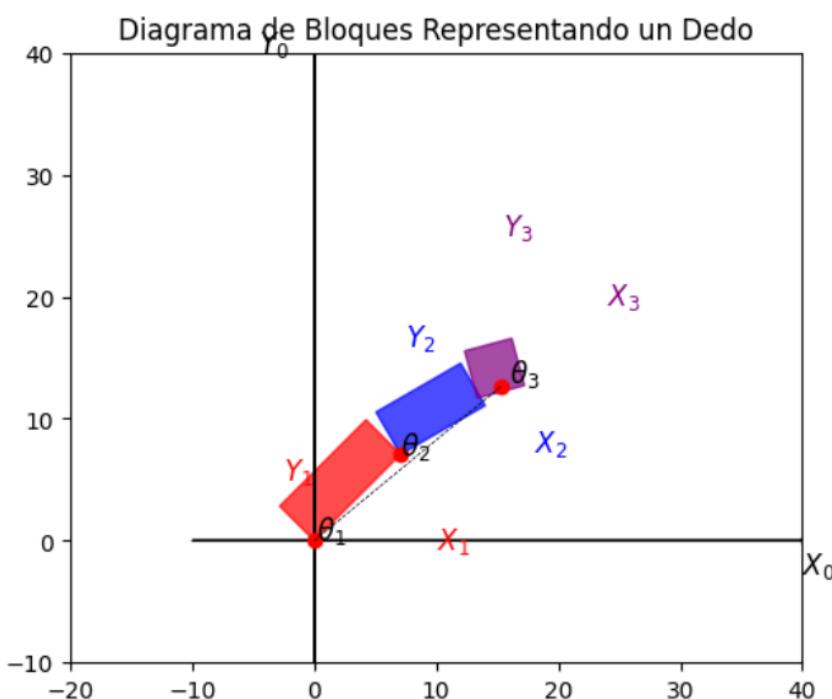


Figura 6.1: Diagrama de bloques simplificando el dedo

De forma que cada barra de distinto color siga un eje de coordenadas de referencia distinto. De esta forma, se simplifican los cálculos al utilizar los datos previos como referencia y no sobre el mismo eje siempre.

Si se utilizase el mismo eje de coordenadas siempre sería más complicado calcular las posiciones que respecto a los ejes que se colocan en cada barra.

Dada la explicación anterior y la anatomía del dedo, el sistema completo de movimiento se puede resumir así:

(No aparece la imagen debido a los derechos de autor. Se recomienda visualizar la imagen: Figure 4: Pulley System Considered) del artículo [19] de las referencias)

En primer lugar, se visualizan los cables de Nylon de flexión y extensión en colores diferentes, conectados por un tubo hueco (en el caso del sistema propio se trata del hueco de la muñeca con los respectivos agujeros hechos para cada cable de cada dedo) y conectados a una polea que en función de la dirección flexione o extienda el dedo.

De esta forma, cada una de las falanges conformará un eje distinto de referencia para el cálculo de posiciones.

6.2. Cálculo de posiciones iniciales

En este apartado se emplean los ejes de coordenadas definidos en el apartado anterior para calcular las posiciones iniciales de los dedos de la mano.

Para ello, se hará uso del estudio de la cinemática directa para obtener las posiciones de cada falange.

Cinemática directa:

Para el cálculo de la cinemática directa que nos de las posiciones de cada falange es necesario tener en cuenta ciertos aspectos teóricos:

Concepto de matriz de rotación: se emplea para calcular posiciones equivalentes entre ejes distintos mediante las funciones trigonométricas de seno y coseno.

La idea es tomar los ejes de coordenadas de referencia iniciales (x_0, y_0) y hallar las posiciones de la primera falange en función de su eje de coordenadas (x_1, y_1).

De tal forma que $x_0 = x_1 \cos \theta_1 - y_1 \sin \theta_1$ y $y_0 = x_1 \sin \theta_1 + y_1 \cos \theta_1$.

Siguiendo esta metodología, se obtiene la matriz de rotación para tres direcciones, la cual es aplicable a la traslación de coordenadas en diferentes ejes:

$$\operatorname{sen}\theta_1 \begin{pmatrix} U(t) \\ V(t) \\ W(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \operatorname{coseno} Rt & -\operatorname{seno} Rt \\ 0 & \operatorname{seno} Rt & \operatorname{coseno} Rt \end{pmatrix} \begin{pmatrix} U(0) \\ V(0) \\ W(0) \end{pmatrix} \operatorname{cos}\theta_1$$

En el caso de la mano, se puede aproximar a un plano, ya que sólo existe movimiento en dos direcciones. En un caso real, se debería tener en cuenta la abducción y aducción de los dedos, que harían de este problema un problema 3D.

Por tanto, se simplifica la matriz anterior a 2D y se fija la dirección Z para los cálculos como fija. Esto nos permitirá usar una matriz simétrica con un parámetro fijo, que será dirección en el espacio.

Para la cinemática directa, tendremos que medir los ángulos, posiciones finales y tener las medidas de las falanges para hallar las posiciones iniciales. Sabiendo la posición final de la falange distal podremos hallar las coordenadas iniciales, sin tener que realizar la conversión de la falange proximal y distal. Cabe añadir que el dedo pulgar sólo tiene dos falanges, no tres como el resto de los dedos, aunque los cálculos se realizan de igual forma.

Un dato que resaltar también es que el pulgar tiene un rango de movimiento mayor a la hora del giro. El eje de referencia sería el mismo, pero teniendo en cuenta la traslación del dedo. Es decir, las coordenadas en 2D serían las mismas, pero en 3D cambiarían por la traslación del dedo unos 90 grados en la flexión extensión completa.

Las medidas se han realizado con instrumentos como un metro o una regla para las longitudes y para los ángulos se ha empleado un transportador de ángulos. Los ángulos que se han tomado como referencia son los que forman las falanges con el interior de la palma de la mano.

Aplicando la matriz anterior a las posiciones finales, ángulos de las tres falanges y la distancia de la falange media y distal se obtienen los siguientes resultados:

Coordenadas de falanges	Pulgar	Índice	Medio	Anular	Meñique
Longitud x2 (cm)	4.5	3	3.5	3	3
Longitud x3 (cm)	2	1.6	2	2	1.5
Falange Distal (x3, y3) (cm)	(0.75, 1)	(0.5, 0.75)	(0.6, 0.9)	(0.5, 0.75)	(0.6, 0.5)
Ángulo α1 (grados)	35	-10	-20	-10	15
Ángulo α2 (grados)	75	-10	-45	-80	-45

Ángulo α_3 (grados)	45	-35	-35	-45	-60
Coordenadas iniciales (x0, y0) (cm)	(2.79,5.28)	(5.35, -1.04)	(4.9, -3.75)	(3.13, -4.4)	(4.69, -4)

Cuadro 6.1: Cinemática del sistema

De esta forma, se obtendrían todas las posiciones iniciales en el estado de reposo (posición del motor =500) y desde la que parten para realizar los ejercicios posteriormente.

Las posiciones en x salen en negativo ya que si se toma como referencia

Cabe añadir que el cálculo de posiciones del pulgar se ha hecho usando la cinemática en el espacio, ya que el pulgar se desplaza en tres direcciones. Para ello, se ha hecho uso de la matriz de rotación anterior al completo.

6.3. Cálculo de velocidades

En este apartado se calculará mediante la cinemática diferencial la velocidad del extremo final del sistema (en este caso la punta del dedo).

6.3.1. Cálculo de velocidades por articulación

Para ello, se debe introducir algunos aspectos teóricos para el cálculo de velocidades:

La jacobiana directa es una herramienta fundamental en la teoría de robots y cinemática. Se utiliza para relacionar las velocidades articulares de un sistema con las velocidades lineales y angulares del extremo del sistema (efector final). La matriz jacobiana es crucial para la planificación de movimientos y el control de cualquier sistema articulado.

Definición de la jacobiana directa:

Para un robot con n grados de libertad (GDL), la jacobiana se define como una matriz que transforma las velocidades articulares en velocidades del extremo del sistema:

Para un sistema con n grados de libertad (GDL), la jacobiana \mathbf{J} se define como una matriz que transforma las velocidades articulares $\dot{\mathbf{q}}$ en velocidades del extremo \mathbf{v} :

$$\mathbf{v} = \mathbf{J}\dot{\mathbf{q}}$$

Donde:

- \mathbf{v} es el vector de velocidades del extremo del sistema, que incluye velocidades lineales y angulares.
- \mathbf{J} es la matriz jacobiana.
- $\dot{\mathbf{q}}$ es el vector de velocidades articulares.

La jacobiana se puede descomponer en dos bloques:

1. **Jacobiana lineal:** Relaciona las velocidades articulares con las velocidades lineales del extremo.
2. **Jacobiana angular:** Relaciona las velocidades articulares con las velocidades angulares del extremo.

Figura 6.2: Captura del código en LaTeX de la definición de jacobiana

Ejemplo de un sistema con 3 GDL:

Consideremos un sistema con 3 grados de libertad (rotacionales), donde las variables articulares son θ_1 , θ_2 y θ_3 . Supongamos que la posición del extremo del sistema en coordenadas cartesianas es $\mathbf{p} = [x, y, z]^T$.

La matriz jacobiana para este sistema puede expresarse como:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial \theta_3} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial \theta_3} \\ \frac{\partial z}{\partial \theta_1} & \frac{\partial z}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} \\ \frac{\partial \theta}{\partial \theta_1} & \frac{\partial \theta}{\partial \theta_2} & \frac{\partial \theta}{\partial \theta_3} \\ \frac{\partial \phi}{\partial \theta_1} & \frac{\partial \phi}{\partial \theta_2} & \frac{\partial \phi}{\partial \theta_3} \\ \frac{\partial \psi}{\partial \theta_1} & \frac{\partial \psi}{\partial \theta_2} & \frac{\partial \psi}{\partial \theta_3} \end{bmatrix}$$

Aquí, ϕ , θ y ψ representan los ángulos de orientación del efecto final.

Figura 6.3: Captura del código en LaTeX de la matriz jacobiana

Sin embargo, hallar las velocidades angulares de cada articulación para posteriormente hallar la del extremo final es muy complicado. En nuestro sistema, es mucho más sencillo hallar la velocidad del extremo final y posteriormente la de las articulaciones teniendo los datos de longitudes y ángulos necesarios.

Para ello, se hará uso de la Jacobiana angular o inversa. A continuación, se encuentra la explicación escrita en Latex para mostrar correctamente los símbolos:

La relación entre las velocidades del extremo \mathbf{e} y las velocidades articulares $\dot{\mathbf{q}}$ está dada por la matriz jacobiana \mathbf{J} :

$$\mathbf{e} = \mathbf{J} \cdot \dot{\mathbf{q}}$$

Si la matriz jacobiana \mathbf{J} es cuadrada y no singular, podemos encontrar la relación inversa:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1} \cdot \mathbf{e}$$

Sin embargo, si \mathbf{J} no es cuadrada o es singular, no se puede calcular la inversa directamente. En estos casos, se utilizan métodos alternativos como la pseudoinversa de Moore-Penrose.

Figura 6.4: Captura del código en LaTeX de la relación de velocidades

En el caso de que no sea cuadrada la matriz \mathbf{J} , se debe realizar el siguiente procedimiento:

Consideremos un sistema con la siguiente matriz jacobiana:

$$\mathbf{J} = \begin{bmatrix} -1.18 & -0.39 & -0.05 \\ -0.36 & -0.9 & -1.1 \end{bmatrix}$$

La relación entre las velocidades en x e y (\mathbf{e}) y las velocidades articulares ($\dot{\mathbf{q}}$) es:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{J} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

Figura 6.5: Captura de LaTeX de la relación de velocidades articulares

Seguidamente, se desarrolla el producto de la Jacobiana por las velocidades angulares y se desarrolla el sistema de ecuaciones siguiente:

$$x = J_{11}\dot{\theta}_1 + J_{12}\dot{\theta}_2 + J_{13}\dot{\theta}_3 = -1.08 \cdot \dot{\theta}_1 - 0.69 \cdot \dot{\theta}_2 - 0.06 \cdot \dot{\theta}_3$$

$$y = J_{21}\dot{\theta}_1 + J_{22}\dot{\theta}_2 + J_{23}\dot{\theta}_3 = -0.26 \cdot \dot{\theta}_1 - 0.94 \cdot \dot{\theta}_2 - 1.11 \cdot \dot{\theta}_3$$

Figura 6.6: Captura de LaTeX del sistema de ecuaciones

Las soluciones a un sistema con dos ecuaciones y tres incógnitas son infinitas.

Es por ello, que para facilitar los cálculos se sustituye la tercera fila de la matriz Jacobiana por (1 1 1) y se incluye una variable nueva. De esta forma queda una matriz cuadrada de 3x3 y un sistema de ecuaciones de la siguiente forma:

$$\mathbf{e} = \mathbf{J} \cdot \dot{\mathbf{q}}$$

$$\begin{bmatrix} x \\ y \\ \epsilon \end{bmatrix} = \mathbf{J} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} -1.08 & -0.69 & -0.06 \\ -0.26 & -0.94 & -1.11 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \mathbf{J}^{-1} \cdot \begin{bmatrix} x \\ y \\ \epsilon \end{bmatrix}$$

$$x = J_{11}\dot{\theta}_1 + J_{12}\dot{\theta}_2 + J_{13}\dot{\theta}_3 = -1.08 \cdot \dot{\theta}_1 - 0.69 \cdot \dot{\theta}_2 - 0.06 \cdot \dot{\theta}_3$$

$$y = J_{21}\dot{\theta}_1 + J_{22}\dot{\theta}_2 + J_{23}\dot{\theta}_3 = -0.26 \cdot \dot{\theta}_1 - 0.94 \cdot \dot{\theta}_2 - 1.11 \cdot \dot{\theta}_3$$

$$\epsilon = J_{31}\dot{\theta}_1 + J_{32}\dot{\theta}_2 + J_{33}\dot{\theta}_3 = \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3$$

Tres ecuaciones, tres incógnitas: $\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3$. Solución única.

Figura 6.7: Resolución del sistema de ecuaciones

Siguiendo todo lo anterior se podrán hallar las velocidades angulares de cada articulación del sistema (dedo) resolviendo el sistema de ecuaciones.

Cabe añadir que la velocidad que se estime para cada articulación de cada dedo será igual en los movimientos de flexión y extensión completa, variará en el caso del ejercicio de flexión y extensión interfalángica.

En primer lugar, se halla la velocidad del extremo de manera experimental con un cronómetro. Sabiendo los ángulos de cada articulación, posiciones finales del extremo y longitud de las falanges media y distal; además de las velocidades angulares finales de cada dedo se podrán hallar las velocidades de cada articulación.

Se toma como aproximación los ángulos de las falanges proximales de la tabla del apartado anterior para las velocidades angulares de extensión, teniendo en cuenta que se parte de $\pi/2$ radianes hasta esos ángulos de reposo. Los ángulos de reposo coinciden con los de la flexión, por lo que sirven para este apartado también.

Por tanto, las velocidades quedarían así:

Dedo	Velocidad angular flexión (rad/s)	Velocidad angular extensión (rad/s)
Pulgar	0.83	0.56
Índice	1.45	1.08
Medio	1.27	1.19
Anular	0.96	0.79
Meñique	0.92	0.72

Cuadro 6.2: Velocidades angulares

Según las especificaciones del motor, tiene una velocidad de 60 grados/0.19s o lo que es lo mismo, 5.2 rad/s. Por tanto, los cálculos entran dentro de los límites de capacidad del motor, viéndose bastante reducida por la transmisión del movimiento de los cables y el peso de los dedos.

Por último, se hallan las velocidades relativas de cada articulación mediante un script de Matlab donde conociendo los ángulos de cada articulación, posiciones finales del extremo y longitud de las falanges media y distal; además de las velocidades angulares finales de cada dedo se podrán hallar las velocidades de cada articulación.

En esta tabla se indican las velocidades medias relativas calculadas, hallando las de flexión y las de extensión y haciendo la media:

Dedo	Velocidad angular falange proximal (rad/s)	Velocidad media falange media (rad/s)	Velocidad angular falange distal (rad/s)
Pulgar	0.41	0.42	-
Índice	0.48	0.48	0.49
Medio	0.42	0.42	0.43
Anular	0.30	0.32	0.34
Meñique	0.30	0.31	0.31

Cuadro 6.3: Velocidades angulares

7. Pruebas y Resultados

7.1. Pruebas manuales

7.1.1. Prueba ID de los servomotores

En esta prueba, se ha realizado la comprobación del cambio de ID de forma que cada motor colocado se corresponda con el dígito de la lista generada en la captura de gestos, y con el dedo correspondiente.

Para ello, se ha hecho uso de la herramienta de la librería externa del GUI mencionado en apartados anteriores.

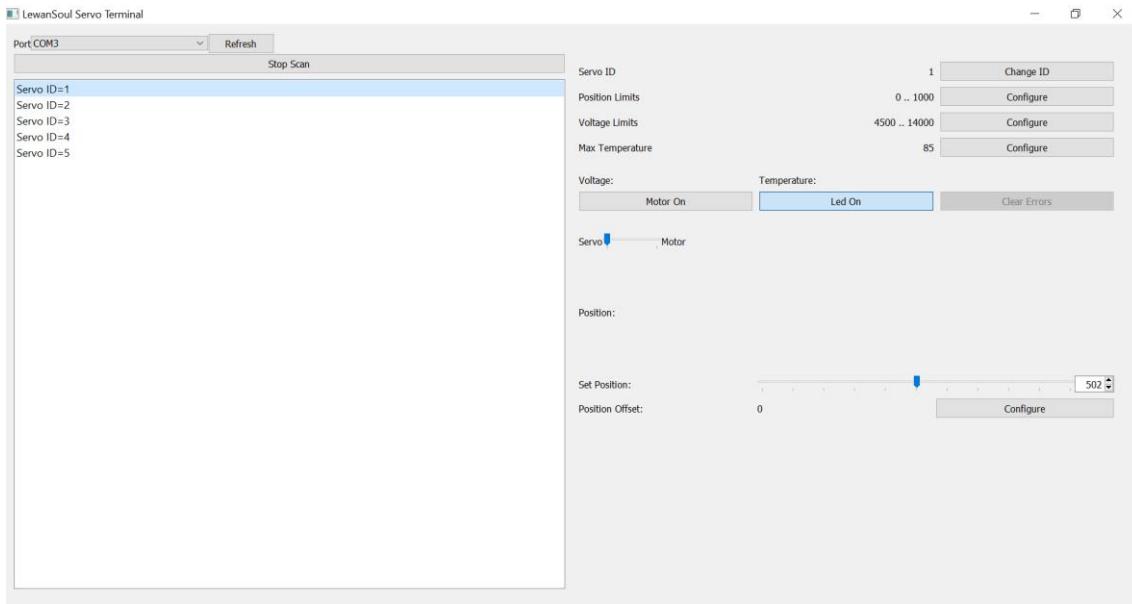


Figura 7.1: Captura del GUI

En la interfaz de usuario anterior existe la opción de cambiar el ID:

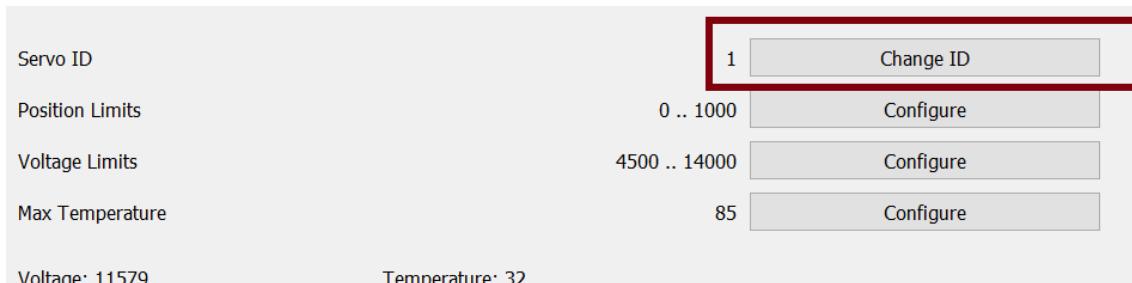


Figura 7.2: Captura del GUI del cambio de ID

Al modificar el ID se debe modificar el motor correspondiente. La idea es que cada motor sea igual al valor de la posición de cada dígito de la lista generada por la captura de gestos. Al no poder comenzar en 0 el ID del motor, el primero debe corresponderse con el 1.

Esta prueba se realizó principalmente porque por defecto todos los motores vienen con el mismo ID, por lo que al conectarlos en serie y abrir el GUI no se detectaba ninguno.

Por tanto, se procedió a analizar uno a uno los motores para ver qué ID tenían y cambiarlos, para que al conectarlos en serie fuesen detectados al tener un ID diferente cada motor.

En la librería del GUI se incluye el ID de broadcast, el cual corresponde al 254. Este ID se encarga de poner todos los motores con el mismo ID también, por lo que nos sirvió para comprobar que al ejecutar el método `.setID(254)` se asignaba un ID aleatorio entre 0-253.

7.1.2. Prueba posiciones servomotores

Empleando el GUI anterior, se pudieron ajustar las posiciones utilizadas en el código para el método `.moveServo()` dependiendo de cada motor y cada dedo.

Dependiendo del dedo, de la posición del motor y de las articulaciones del dedo se ajustaron las posiciones.

Durante el montaje de la mano, se limaron las piezas en función de cómo habían sido impresas y cómo encajaban unas con otras, por lo que el resultado dio que algunos dedos son más sencillos de mover y requieren un menor torque del motor, y otros dedos necesitan un mayor torque al estar más compactos y ser más difícil rotar sus articulaciones.

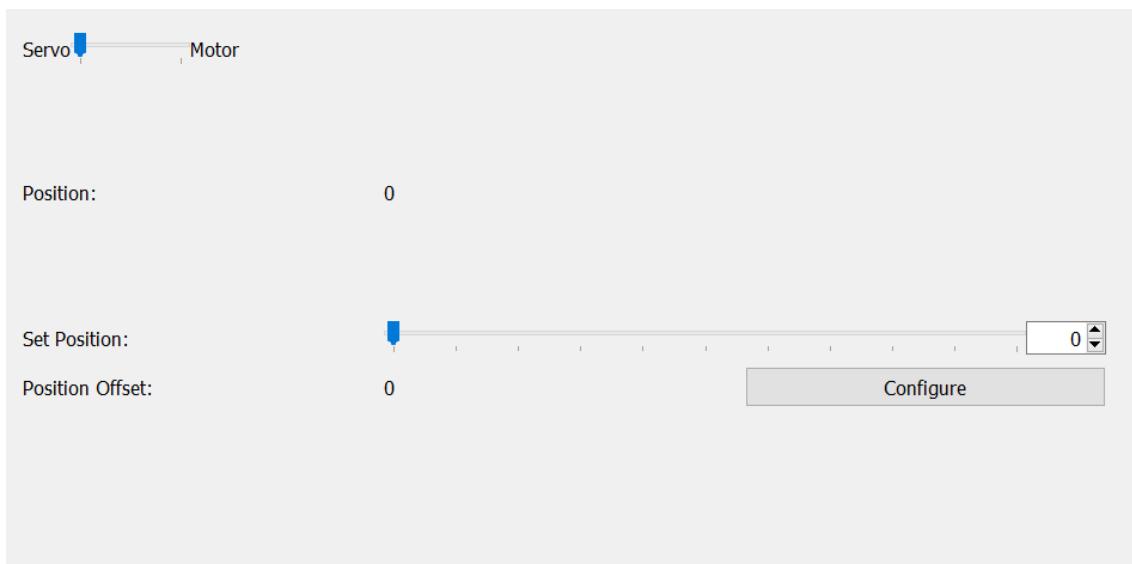


Figura 7.3: Captura del GUI del cambio de posición

Siempre en modo de **Servo** se regula la posición y se ha ido probando para ajustar la de cada dedo, en función del ejercicio y el movimiento que se requiera.

Si se ejecutase en modo **Motor** se podría ajustar la velocidad, no la posición.

Por último, de esta forma se ha seleccionado 500 como el punto de partida de reposo para todos los dedos.

7.1.3. Prueba sobre las articulaciones de los dedos

Para esta prueba, se ha hecho uso de los diferentes materiales empleados para las articulaciones, comprobando cuál funciona mejor para cada articulación.

Los dedos, al extenderse, deben tener una articulación sin holgura, dado que si la articulación se mueve puede impedir el movimiento del dedo y lo mantendría fijo.

Por tanto, se han seleccionado diferentes materiales mediante prueba y error con el GUI anterior para cada falange.

Las falanges distales de dedo medio, anular y meñique están unidas por una cuerda blanca de grosor 1mm aproximado. Aunque se trate de una cuerda elástica impide que haya holgura en el movimiento al ser uniones de un grosor muy pequeño.

La del índice y la del pulgar están formadas por la cuerda gris más rígida y de menor grosor, dado que las piezas encajan de manera más compacta entre sí y no hay apenas holgura. Por tanto, haciendo de soporte es suficiente para que se realice la extensión de los dedos.

El resto de las falanges se han unido con cable de 1mm de grosor, el cual es más rígido e impide que haya holgura en ningún tipo de movimiento.

7.2. Resultados

Los resultados obtenidos han servido para estimar las posiciones a las que deben moverse los servomotores LX16A para completar los movimientos.

Además, mediante esta herramienta se ha conseguido realizar todas las pruebas con éxito, comprobando posteriormente los ID, posición del motor tras restablecer posición con la función **set_motors()** o visualmente comprobar que los dedos se mueven de manera correcta en función de lo que se busque en cada movimiento.

8. Análisis temporal y viabilidad

8.1. Análisis temporal

En este apartado se comparan las estimaciones del apartado 2 con el resultado final de la distribución de cada tarea. Para ello, se estableció una línea base inicial y se compara con la real.

En primer lugar, se muestra la nueva distribución temporal de las tareas:

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Búsqueda de información sobre la temática del TFG	15 días	lun 16/10/23	vie 03/11/23	
Búsqueda de documentación sobre motores y modelo 3D	59 días?	mié 25/10/23	lun 15/01/24	
Poner en funcionamiento los motores	35 días?	lun 15/01/24	vie 01/03/24	
Estudio transmisión movimiento	13 días	vie 01/03/24	mar 19/03/24	
Impresión 3D de las piezas	44 días?	vie 01/03/24	mié 01/05/24	
Ensamblado de las piezas 3D	17 días?	lun 15/04/24	mar 07/05/24	

Montaje estructura	23 días?	lun 15/04/24	mié 15/05/24	
Afinar funcionamientos motores	23 días?	lun 15/04/24	mié 15/05/24	
Justificación clínica	5 días?	lun 15/04/24	dom 21/04/24	
Pruebas de los motores	24 días?	jue 18/04/24	mar 21/05/24	
Cálculo de cinemática	5 días?	lun 20/05/24	sáb 25/05/24	6
Memoria del proyecto	10 días?	mié 15/05/24	mar 28/05/24	
Fin TFG	1 día	mar 28/05/24	mar 28/05/24	

Cuadro 8.1: Planificación real

Principalmente, se debe recalcar que la impresión de las piezas 3D se retrasó bastante más de lo esperado, por lo que atrasó todo el proyecto. De esta forma, se comprimieron mucho las tareas durante las últimas dos semanas.

Aquí, se puede visualizar el Diagrama de Gantt de la planificación real:

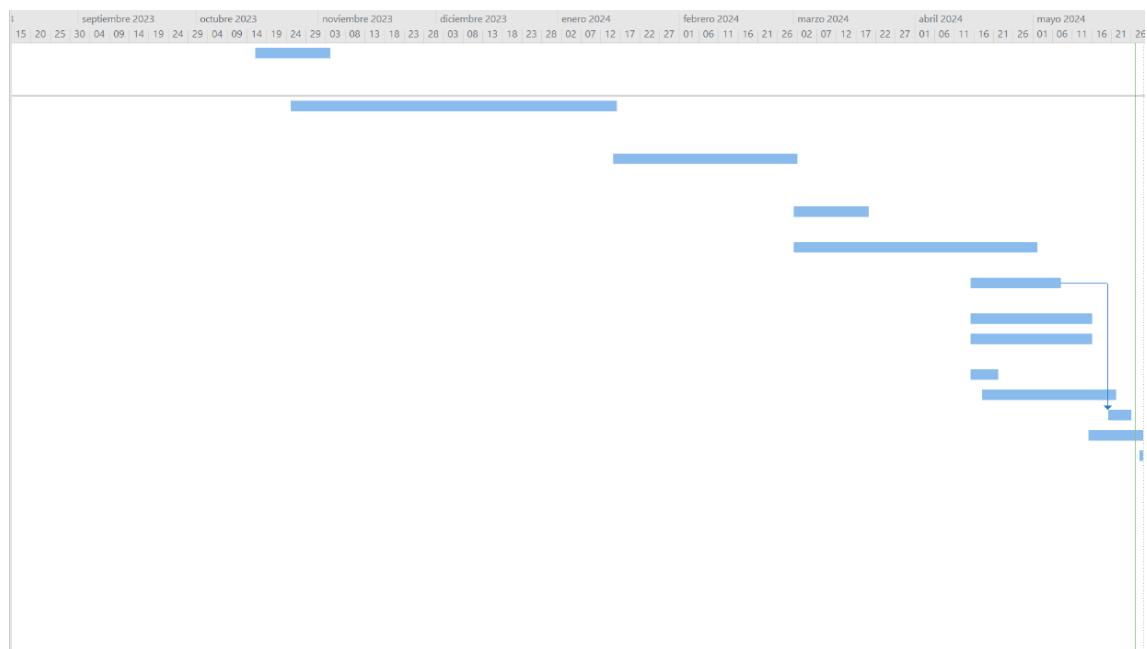


Figura 8.1: Diagrama de Gantt real

Y así quedaría la comparativa entre la línea base estimada y la real:

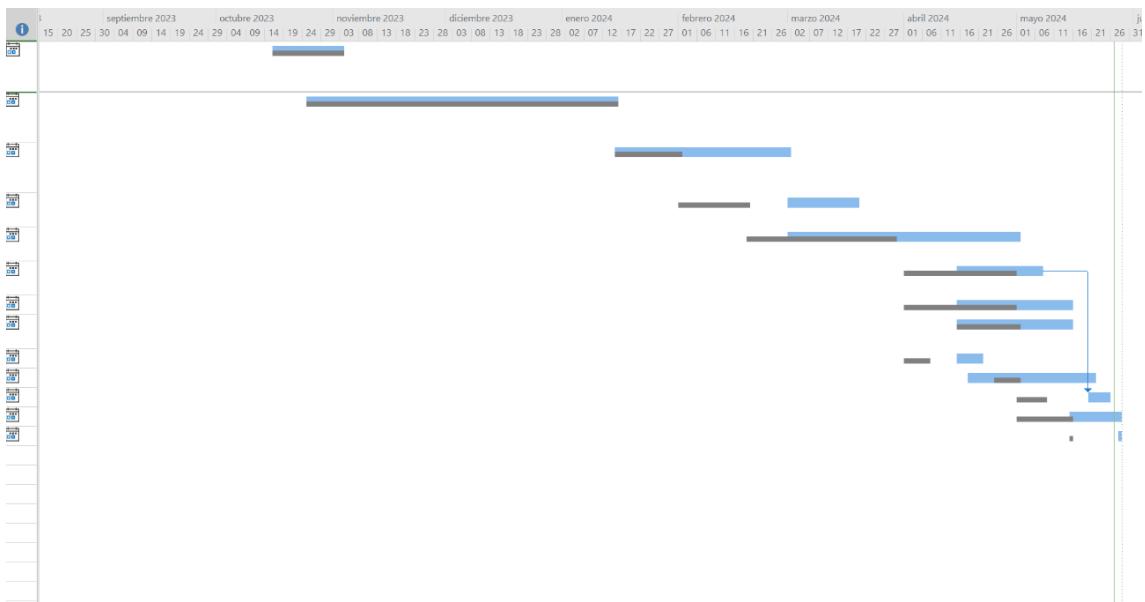


Figura 8.2: Comparativa del Diagrama de Gantt real y el estimado

8.2. Viabilidad y desarrollo futuro

Dado el bajo coste de realización del proyecto se puede plantear la implantación de estos sistemas de rehabilitación en el mayor número de clínicas posibles.

Por un lado, se podría desarrollar este producto para que incluya algún tipo de feedback de carácter terapéutico que permita hacer los ejercicios monitorizados en tiempo real a los pacientes, a la vez que el fisioterapeuta controle visualmente los movimientos en su consulta.

Por otro lado, la idea de introducir una base de datos que almacene la lista actualizada de la posición de los dedos sería un avance importante. Tener estos datos permitiría al especialista monitorizar la evolución del paciente en cualquier momento, sin necesidad de ser en tiempo real. El especialista ejecutaría el software que se encargue de mover los motores en función de los valores recibidos y se introducirían los valores de la base de datos como parámetros.

Con todo lo anterior se crearía una herramienta de monitorización con feedback y a tiempo real o en diferido mucho más completa, que permita introducir alguna innovación en las herramientas digitales del mundo de la tele rehabilitación.

Por último, las librerías de captura de imágenes permiten leer mucha más información que la posición de los dedos con 1 y 0 como se ha realizado en este proyecto.

Hay métodos de la librería Mediapipe que permiten obtener las coordenadas de cada punto que aparece en la captura de imágenes como los del ejemplo siguiente:

```
1 lmList1 = hand1["lmList"]# List of 21 landmark points
2         bbox1 = hand1["bbox"] #Bounding box info x,y,w,h
3         centerPoint1 = hand1["center"] #center of the hand, cx
4         and cy
```

Código 8.1: Métodos librería Mediapipe

Los métodos anteriores escritos en el lenguaje Python permiten obtener coordenadas mucho más complejas de la mano. De esta forma se podría filtrar de manera más eficiente la información y afinar la mano de manera que se mueva a tiempo real dependiendo de las coordenadas captadas y simule el movimiento de la mano a tiempo real.

También, todo ello serviría para impulsar la realimentación o feedback de los ejercicios realizados de manera mucho más precisa y no tan simplificada.

9. Conclusiones

El desarrollo de sistemas biomecatrónicos para la rehabilitación de la mano aporta varios beneficios importantes al campo de la fisioterapia. Este proyecto demostró que es posible desarrollar herramientas que permitan a los fisioterapeutas monitorizar y evaluar los ejercicios de rehabilitación realizados por los pacientes en casa en tiempo real. La integración de tecnologías como los servomotores y la impresión 3D fue esencial para reproducir con precisión los movimientos de la mano y garantizar la eficacia del tratamiento.

Los logros clave incluyen el desarrollo de una mano robótica que puede realizar movimientos de rehabilitación específicos, aumentar el cumplimiento de los programas de rehabilitación y mejorar la precisión de los movimientos realizados por los pacientes. Además, el sistema proporciona la flexibilidad necesaria para atender a pacientes con movilidad limitada, lo que permite una mejor integración y acceso al tratamiento.

Este proyecto también enfatiza la importancia de la colaboración interdisciplinaria, combinando conocimientos de ingeniería biomédica, electrónica y de control con una comprensión de las necesidades clínicas específicas. Las mejoras futuras incluyen la implementación de algoritmos de inteligencia artificial más avanzados para capturar y analizar mejor el movimiento, así como ampliar el sistema para cubrir una gama más amplia de patologías de las manos.

En conclusión, la construcción de un sistema de rehabilitación híbrido no sólo optimiza los recursos y el tiempo de espera de las clínicas de fisioterapia, sino que también proporciona una alternativa viable y eficaz para la rehabilitación de pacientes con limitaciones de movilidad.

10. Referencias

- [1] Abdelhafiz, M. H., Spaich, E. G., Dosen, S., & Andreasen Struijk, L. N. S. (2019). Bio-inspired tendon driven mechanism for simultaneous finger joints flexion using a soft hand exoskeleton. *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, 1073-1078. doi:10.1109/ICORR.2019.8779547
- [2] Bahrami, S., & Dumond, P. (2018). Testing of coiled nylon actuators for use in spastic hand exoskeletons. *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 1853-1856. doi:10.1109/EMBC.2018.8512596
- [3] Bachtiar, Y., Pristovani, R. D., Dewanto, S., & Pramadihanto, D. (2018). Mechanical and forward kinematic analysis of prosthetic robot hand for T-FLoW 3.0. *2018 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, 275-280. doi:10.1109/ELECSYM.2018.8615522
- [4] Carothers, W. H. (1938). Polyamides and their production. United States Patent No. 2,130,948.
- [5] Craig, J. J. (2005). *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall.
- [6] DuPont. (n.d.). The history of nylon. DuPont. Recuperado de <https://www.dupont.com/knowledge/history-of-nylon.html>
- [7] Harris, J. E., Eng, J. J., Miller, W. C., & Dawson, A. S. (2009). A self-administered Graded Repetitive Arm Supplementary Program (GRASP) improves arm function during inpatient stroke rehabilitation: a multi-site randomized controlled trial. *Stroke*, 40*(6), 2123-2128. doi:10.1161/STROKEAHA.108.544585
- [8] Heo, P., & Kim, J. (2012). Finger flexion force sensor based on volar displacement of flexor tendon. *2012 IEEE International Conference on Robotics and Automation*, 1392-1397. doi:10.1109/ICRA.2012.6224818
- [9] Hrabar, I., Čelan, B., Matić, D., Jerković, N., & Kovačić, Z. (2021). Towards supervised robot-assisted physical therapy after hand fractures. *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 1-6. doi:10.23919/SoftCOM52868.2021.9559079
- [10] Krupp, L. B., LaRocca, N. G., Muir-Nash, J., & Steinberg, A. D. (1989). The fatigue severity scale. Application to patients with multiple sclerosis and systemic lupus erythematosus. *Arch Neurol*, 46*(10), 1121-1123. doi:10.1001/archneur.1989.00520460115022
- [11] Khatik, V., & Saxena, A. (2024). On optimal tendon routing-based design of biologically inspired underactuated hand exoskeleton for gross grasping. *IEEE Transactions on Medical Robotics and Bionics*, 6*(2), 600-617. doi:10.1109/TMRB.2024.3387334
- [12] Mnyusiwalla, H., Vulliez, P., Gazeau, J. -P., & Zeghloul, S. (2016). A new dexterous hand based on bio-inspired finger design for inside-hand manipulation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46*(6), 809-817. doi:10.1109/TSMC.2015.2468678

- [13] Morton, W. E., & Hearle, J. W. S. (2008). *Physical Properties of Textile Fibres* (4^a ed.). Woodhead Publishing.
- [14] Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Springer.
- [15] Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot Modeling and Control*. Wiley.
- [16] Treratanakulwong, T., Kaminaga, H., & Nakamura, Y. (2014). Low-friction tendon-driven robot hand with carpal tunnel mechanism in the palm by optimal 3D allocation of pulleys. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 6739-6744. doi:10.1109/ICRA.2014.6907854
- [17] Williams, D. J., Krebs, H. I., & Hogan, N. (2001). A robot for wrist rehabilitation. *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1336-1339. doi:10.1109/IEMBS.2001.1020443
- [18] Zabarauskas, M., & Cameron, S. (2014). Luke: An autonomous robot photographer. *Proceedings - IEEE International Conference on Robotics and Automation*, 1809-1815. doi:10.1109/ICRA.2014.6907096
- [19] Zhang, X. (2018). Designing humanoid robot controllers for physical human-robot interaction. *Murray State Theses and Dissertations*. Recuperado de <https://digitalcommons.murraystate.edu/cgi/viewcontent.cgi?article=1171&context=honorstheses>
- [20] Autodesk. (n.d.). Fusion 360. Recuperado de <https://www.autodesk.es/products/fusion-360/overview?term=1-YEAR&tab=subscription>
- [21] Hiwonder. (n.d.). LX-16A. Recuperado de <https://www.hiwonder.com/products/lx-16a>
- [22] Mediapipe by Google. (n.d.). Recuperado de <https://ai.google.dev/edge/mediapipe/solutions/guide?hl=es-419>
- [23] Mediapipe Hand Landmarker. (n.d.). Recuperado de https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker?hl=es-419
- [24] Robotis. (n.d.). USB2Dynamixel. Recuperado de <https://emanual.robotis.com/docs/en/parts/interface/usb2dynamixel/>
- [25] Ethan Lipson. (n.d.). PyLX-16A. GitHub. Recuperado de <https://github.com/ethanlipson/PyLX-16A.git>
- [26] Maxim Kulkin. (n.d.). LewanSoul LX-16A. GitHub. Recuperado de <https://github.com/maximkulkin/lewansoul-lx16a.git>
- [27] Jupyter. (n.d.). Jupyter Notebook. Recuperado de <https://jupyter.org/>
- [28] OpenCV. (n.d.). Recuperado de <https://opencv.org/>
- [29] Rehand. (n.d.). Inicio. Recuperado de <https://rehand.net/es/inicio/>

- [30] IEEE Xplore. (n.d.). Mechanical and Forward Kinematic Analysis of Prosthetic Robot Hand for T-FLoW 3.0. Recuperado de <https://ieeexplore.ieee.org/document/8615522>
- [31] Elsevier. (n.d.). Abstract of S0894-1130(20)30186-1. Recuperado de [https://www.jhandtherapy.org/article/S0894-1130\(20\)30186-1/abstract](https://www.jhandtherapy.org/article/S0894-1130(20)30186-1/abstract)
- [32] AllDatasheet. (n.d.). 74LS241N. Recuperado de <https://www.alldatasheet.com/view.jsp?Searchword=74LS241N&sField=3>
- [33] AllDatasheet. (n.d.). 74HC126N. Recuperado de <https://www.alldatasheet.es/datasheet-pdf/pdf/15536/PHILIPS/74HC126N.html>
- [34] Overleaf. Recuperado de <https://www.overleaf.com/>